SBRC 2016

XXXIV Simpósio Brasileiro
de Redes de Computadores
e Sistemas Distribuídos

30 de maio a 03 de junho de 2016
Salvador - Bahia - Brasil
www.sbrc2016.ufba.br

Foto Pierre Verger ©Fundação Pierre Verger

ANAIS **SBRC 2016**

REALIZAÇÃO

computação UFBA

UFBA 70 ANOS

INSTITUTO FEDERAL BAHIA

XXXIV Simpósio Brasileiro
de Redes de Computadores
e Sistemas Distribuídos

SBRC 2016

30 de maio a 03 de junho de 2016
Salvador - Bahia - Brasil
www.sbrc2016.ufba.br

Anais do SBRC 2016
Trilha Principal e Salão de Ferramentas

Editora

Sociedade Brasileira de Computação (SBC)

Organização

Antonio Marinho Pilla Barcellos (UFRGS)
Antonio Alfredo Ferreira Loureiro (UFMG)
Fabíola Gonçalves Pereira Greve (UFBA)
Allan Edgard Silva Freitas (IFBA)

Realização

Universidade Federal da Bahia (UFBA)
Instituto Federal da Bahia (UFBA)

Promoção

Sociedade Brasileira de Computação (SBC)
Laboratório Nacional de Redes de Computadores (LARC)

Capa: Gilson Rabelo (UFBA)
Produção Editorial: Antonio Augusto Teixeira Ribeiro Coutinho (UEFS)

Cópias Adicionais:
Sociedade Brasileira de Computação (SBC)
Av. Bento Gonçalves, 9500- Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia - CEP 91.509-900 - Porto Alegre - RS
Fone: (51) 3308-6835
E-mail: sbc@sbc.org.br

Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (34: 2016: Salvador, BA).

Anais / XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos; organizado por Antonio Marinho Pilla Barcellos, Antonio Alfredo Ferreira Loureiro, Daniel Macêdo Batista, Fabíola Gonçalves Pereira Greve, Allan Edgard Silva Freitas - Porto Alegre: SBC, 2016

1254 p. il. 21 cm.
Vários autores
Inclui bibliografias
ISSN: 2177-496X

1. Redes de Computadores. 2. Sistemas Distribuídos. I. Barcellos, Antonio Marinho Pilla II. Loureiro, Antonio Alfredo Ferreira III. Batista, Daniel Macêdo IV. Greve, Fabíola Gonçalves Pereira V. Freitas, Allan Edgard Silva VI. Título.

Sociedade Brasileira da Computação

Contato
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS
CNPJ: 29.532.264/0001-78
http://www.sbrc.org.br

Laboratório Nacional de Redes de Computadores (LARC)

Diretora do Conselho Técnico-Científico
Rossana Maria de C. Andrade (UFC)

Vice-Diretor do Conselho Técnico-Científico
Ronaldo Alves Ferreira (UFMS)

Diretor Executivo
Paulo André da Silva Gonçalves (UFPE)

Vice-Diretor Executivo
Elias Procópio Duarte Jr. (UFPR)

Membros Institucionais
SESU/MEC, INPE/MCT, UFRGS, UFMG, UFPE, UFCG (ex-UFPB Campus Campina Grande), UFRJ, USP, PUC-Rio, UNICAMP, LNCC, IME, UFSC, UTFPR, UFC, UFF, UFSCar, IFCE (CEFET-CE), UFRN, UFES, UFBA, UNIFACS, UECE, UFPR, UFPA, UFAM, UFABC, PUCPR, UFMS, UnB, PUC-RS, UNIRIO, UFS e UFU.

Contato
Universidade Federal de Pernambuco - UFPE
Centro de Informática - CIn
Av. Jornalista Anibal Fernandes, s/n
Cidade Universitária
50.740-560 - Recife - PE
http://www.larc.org.br

Organização do SBRC 2016

Coordenadores Gerais
Fabíola Gonçalves Pereira Greve (UFBA)
Allan Edgard Silva Freitas (IFBA)
Romildo Martins da Silva Bezerra (IFBA) - In Memoriam

Coordenadores do Comitê de Programa
Antonio Marinho Pilla Barcellos (UFRGS)
Antonio Alfredo Ferreira Loureiro (UFMG)

Coordenador de Palestras e Tutoriais
Francisco Vilar Brasileiro (UFCG)

Coordenador de Painéis e Debates
Carlos André Guimarães Ferraz (UFPE)

Coordenadores de Minicursos
Lau Cheuck Lung (UFSC)
Frank Siqueira (UFSC)

Coordenadora de Workshops
Michele Nogueira Lima (UFPR)

Coordenador do Salão de Ferramentas
Daniel Macêdo Batista (USP)

Comitê de Organização Local
Adolfo Almeida Duran (UFBA)
Allan Edgard Silva Freitas (IFBA)
Antonio Augusto Teixeira Ribeiro Coutinho (UEFS)
Cátia Mesquita Brasil Khouri (UESB)
Fabíola Gonçalves Pereira Greve (UFBA)
Flávia Maristela Nascimento (IFBA)
Gustavo Bittencourt Figueiredo (UFBA)
Ítalo Valcy da Silva Brito (UFBA)
Jauberth Weyll Abijaude (UESC)
Manoel Carvalho Marques Neto (IFBA)
Marco Antônio Dantas Ramos (UESB)
Marcos Yuzuru de Oliveira Camada (IFBaiano)
Marcos Ennes Barreto (UFBA)
Maycon Leone Maciel Peixoto (UFBA)
Rafael Freitas Reale (IFBA)
Renato Lima Novais (IFBA)
Ricardo Araújo Rios (UFBA)
Romildo Martins da Silva Bezerra (IFBA) - In Memoriam
Sandro Santos Andrade (IFBA)
Vinícius Tavares Petrucci (UFBA)

Mensagem dos Coordenadores Gerais

Sejam bem-vindos ao 34o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) e à nossa terra, Salvador da Bahia.

O desafio de organizar um evento do porte, escopo e excelência do SBRC é substancial. Agradecemos à comunidade pela confiança e privilégio em poder contribuir para a mais tradicional e relevante conferência científica nacional na sua área e uma das mais concorridas da Ciência da Computação na América Latina. No contexto desafiador em que o país se encontra nesse ano de 2016, é ainda mais importante reforçar o papel primordial do SBRC como fórum de discussão, encontros e divulgação dos melhores trabalhos produzidos pela comunidade de pesquisa nacional.

A programação do SBRC 2016 está diversificada, abrangente e tem excelente qualidade; nesse sentido, a contribuição da comunidade foi fundamental para a valorização do evento e o fortalecimento da Ciência e Tecnologia no nosso país. A programação engloba 27 sessões técnicas com apresentação de 81 artigos científicos completos, selecionados por meio de um rigoroso trabalho de revisão, 3 palestras proferidas por pesquisadores internacionalmente renomados, e 3 painéis de discussões e debates, todos em temas atuais, como Computação em Nuvem, Internet da Coisas, Cidades Inteligentes e Experimentação. São oferecidos 6 minicursos, voltados à formação e atualização dos participantes em temas de ponta, e apresentados 2 tutoriais, de forma a aproximar o público dos avanços tecnológicos.

Adicionalmente, há a exposição de 8 ferramentas no Salão de Ferramentas e realização de 8 workshops, em paralelo ao SBRC, com foco em temas de pesquisa e desenvolvimento específicos e emergentes. Nesta edição, através do prêmio "Destaque SBRC", também homenageamos uma personalidade nas áreas de Redes de Computadores e Sistemas Distribuídos, pela sua importante contribuição para o fomento da pesquisa e estruturação de uma sólida comunidade científica no Brasil.

A excelência das atividades programadas nesta edição é reflexo da competência e empenho dos seus respectivos coordenadores. Um agradecimento muito especial a Marinho Barcellos (UFRGS), Antonio Alfredo Ferreira Loureiro (UFMG), Francisco Vilar Brasileiro (UFCG), Carlos André Guimarães Ferraz (UFPE), Lau Cheuck Lung (UFSC), Frank Siqueira (UFSC), Michele Nogueira Lima (UFPR) e Daniel Macêdo Batista (USP). Exaltamos ainda o trabalho voluntário, intenso, atencioso e contínuo, realizado pelos colegas do Comitê de Organização Local e através deles agradecemos o apoio das instituições parceiras: Universidade Estadual de Feira de Santana (UEFS), Universidade Estadual do Sudoeste da Bahia (UESB), Universidade Estadual de Santa Cruz (UESC) e Instituto Federal Baiano (IFBaiano).

Agradecemos às diretorias da SBC e do LARC, promotores do SBRC, pela confiança depositada em nós, e pelo competente apoio organizacional prestado pela equipe administrativa da SBC. Somos também gratos aos integrantes do Comitê Consultivo do SBRC e à coordenação da Comissão Especial de Redes de Computadores e Sistemas Distribuídos da SBC, pelos aconselhamentos e pelo suporte financeiro inicial prestado à organização do SBRC 2016. Gostaríamos ainda de agradecer aos patrocinadores do simpósio: ao Comitê Gestor da Internet no Brasil (CGI.br), aos órgãos de fomento, CNPq, CAPES, e ACM Sigcomm, aos nossos apoiadores e às empresas patrocinadoras por valorizarem e reconhecerem o SBRC como um evento importante para o fomento à pesquisa e inovação. Nosso agradecimento especial às nossas instituições, especialmente, ao Departamento de Ciência da Computação, ao Instituto de Matemática e à Reitoria da UFBA, e ao Departamento de Computação, ao Campus de Salvador e à Reitoria do IFBA, pelo indispensável suporte para a realização do SBRC.

Por fim, sensibilizados com a perda precoce do nosso amigo e colega Romildo Martins

da Silva Bezerra (IFBA), organizador in memoriam dessa edição, gostaríamos de deixar o nosso testemunho da sua grandeza, energia e exemplo de profissional e pesquisador.

Em nome do Comitê Organizador do SBRC 2016, desejamos a todos uma semana agradável em Salvador, rica em discussões e encontros.

Fabíola Greve (UFBA) e Allan Freitas (IFBA)
Coordenadores Gerais do SBRC 2016

Mensagem dos Coordenadores do Comitê de Programa do SBRC 2016

O SBRC é um evento científico nacional consolidado e de reconhecida qualidade. Seu histórico de submissões reflete a força da comunidade que realiza pesquisa em redes de computadores e sistemas distribuídos. Na sua 34ª edição, foram submetidos 229 trabalhos completos, dos quais 81 foram aceitos para apresentação e publicação, correspondente a 35% de taxa de aceitação.

Todos os 229 trabalhos passaram por uma avaliação criteriosa feita pelos 122 membros do Comitê de Programa e por 103 revisores associados a eles. Cada artigo foi avaliado por, pelo menos, três especialistas na área, sendo que a grande maioria dos trabalhos teve quatro pareceres. Como tem ocorrido há várias edições do SBRC, após a etapa inicial de avaliação dos trabalhos, os autores tiveram a oportunidade de submeter uma rebuttal (réplica), fazendo os devidos esclarecimentos técnicos. Em seguida, cada revisor foi incentivado a avaliar o rebuttal e discutir as avaliações com os demais revisores. Nesse processo, contamos com a ajuda de revisores líderes, membros mais experientes no comitê de programa, responsáveis por convergir a discussão e chegar a uma recomendação final. Especial atenção foi dada aos artigos na "zona cinza", ou seja, onde havia dúvidas sobre aceitação ou rejeição. Ao discutir o ponto de corte, os membros do comitê foram consultados sobre a taxa de aceitação. Por uma questão de transparência, os trabalhos submetidos pelos coordenadores do comitê de programa foram avaliados de forma totalmente independente (externo ao JEMS), e agregados apenas após a definição do ponto de corte. Finalizado todo o processo de seleção, os resultados foram consolidados, discutidos e homologados pelos membros do Comitê de Programa. Posteriormente, foi organizada uma comissão formada por alguns membros do Comitê de Programa para escolher os trabalhos destaque desta edição do evento.

O resultado final é uma programação técnica que reflete muito bem a diversidade e o vigor das iniciativas de pesquisa em curso nas universidades, centros de pesquisa e empresas do País.

Gostaríamos de expressar os nossos sinceros agradecimentos aos membros do Comitê de Programa e revisores pelo árduo trabalho feito e pela pronta resposta às nossas várias chamadas e solicitações. A grande maioria dos pareceres foi de excelente qualidade, demonstrando grande objetividade, imparcialidade e com boas sugestões para os autores. A publicação de um trabalho científico só se realiza com a ativa participação de revisores sérios e comprometidos com esse processo. Somente com esse trabalho cuidadoso de avaliação e seleção de artigos que o alto nível de qualidade do SBRC pode ser mantido.

Queremos fazer um agradecimento especial aos coordenadores gerais deste 34o SBRC – professores Fabíola Gonçalves Pereira Greve (UFBA) e Allan Edgard Silva Freitas (IFBA) – pela confiança depositada em nosso trabalho e pelo apoio durante todo o processo. Agradecemos também aos autores que submeteram trabalhos ao SBRC 2016, fundamentais para seu sucesso.

Desejamos a todos os participantes do SBRC 2016 uma semana bastante produtiva e rica em discussões técnicas instigantes bem como conversas inspiradoras para novos projetos e cooperações.

Marinho Barcellos (UFRGS)
Antonio Alfredo Ferreira Loureiro (UFMG)
Coordenadores do Comitê de Programa

Mensagem do Coordenador do Salão de Ferramentas do SBRC 2016

Bem-vindos ao Salão de Ferramentas do SBRC!

Todos que fazem pesquisa científica em redes de computadores e sistemas distribuídos sabem da importância de boas ferramentas de software disponíveis. Além de reduzir o intervalo de tempo entre a ideia do projeto e a obtenção dos resultados, considerando claro que as ferramentas sejam bem feitas e bem documentadas, a utilização de tais ferramentas aumenta a credibilidade no trabalho realizado já que a reprodução de experimentos e a validação dos resultados tornam-se tarefas fáceis de serem realizadas. Vale ressaltar também a importância que as ferramentas tem em ambientes de produção, evitando gastos elevados e perda de tempo na implantação desta ou daquela solução.

Reconhecer o bom trabalho feito no desenvolvimento e disponibilização de ferramentas para redes de computadores e sistemas distribuídos é uma tarefa importante para garantir que esse processo não morra, e por isso fico muito feliz em ter sido convidado para coordenar o Salão de Ferramentas em 2016.

Esta é a primeira vez que faço parte do quadro de coordenadores da organização geral do SBRC. Sem dúvida foi uma ótima experiência, dado o tamanho do evento e também pelo fato desta edição acontecer na minha cidade natal. Agradeço a todas as pessoas que sugeriram meu nome e que defenderam essa minha indicação.

Neste ano 26 ferramentas foram registradas, e destas, 20 foram de fato submetidas. Após uma avaliação comparativa considerando as revisões feitas por 36 pesquisadores, 8 ferramentas foram aceitas para apresentação no Salão. Os principais tópicos dos trabalhos aceitos são redes definidas por software, computação em nuvem, redes ópticas e cidades inteligentes. Agradeço ao excelente trabalho dos 30 membros do comitê de programa durante todo o processo de divulgação do Salão e de definição das ferramentas aceitas.

Finalizo esta mensagem agradecendo às valiosas dicas e ajudas dos coordenadores de salões passados, Alfredo Goldman e Cesar Marcondes, a todo o apoio dos coordenadores gerais do SBRC, Fabíola, Allan e Romildo (in memoriam) e às explicações a respeito da infraestrutura do Salão feitas pelo Marcos Camada da organização local.

Espero que todos aproveitem o Salão de Ferramentas e que as ferramentas apresentadas sejam úteis para a comunidade!

Daniel Macêdo Batista (USP)
Coordenador do Salão de Ferramentas do SBRC 2016

Comitê de Programa da Trilha Principal

- Alberto Egon Schaeffer-Filho (UFRGS)
- Aldri Luiz dos Santos (UFPR)
- Alex Borges Vieira (UFJF)
- Alfredo Goldman (USP)
- Allan Edgard Silva Freitas (IFBA)
- Alysson Neves Bessani (UL - Portugal)
- Ana Paula Couto da Silva (UFMG)
- André Costa Drummond (UnB)
- André Castelo Branco Soares (UFPI)
- Andre L. L. Aquino (UFAL)
- Anelise Munaretto (UTFPR)
- Antonio Alfredo Ferreira Loureiro (UFMG)
- Antonio Augusto de Aragão Rocha (UFF)
- Antônio Jorge Gomes Abelém (UFPA)
- Antonio Marinho Pilla Barcellos (UFRGS)
- Antonio Tadeu Azevedo Gomes (LNCC)
- Artur Ziviani (LNCC)
- Carlos Alberto Kamienski (UFABC)
- Carlos Alberto Vieira Campos (UNIRIO)
- Carlos André Guimarães Ferraz (UFPE)
- Carlos Maurício Seródio Figueiredo (UEA)
- Célio Vinicius Neves de Albuquerque (UFF)
- Cesar Augusto Cavalheiro Marcondes (UFSCar)
- Christian Esteve Rothenberg (UNICAMP)
- Cristiano Bonato Both (UFCSPA)
- Daniel Fernandes Macedo (UFMG)
- Daniel Guidoni (UFSJ)
- Daniel Macêdo Batista (USP)
- Daniel Sadoc Menasche (UFRJ)
- Dênio Mariz Timóteo de Sousa (IFPB)
- Djamel Fawzi Hadj Sadok (UFPE)
- Dorgival Guedes (UFMG)
- Edmundo de Souza e Silva (UFRJ)
- Edmundo Roberto Mauro Madeira (UNICAMP)
- Eduardo Cerqueira (UFPA)
- Eduardo Freire Nakamura (UFAM)
- Elias Procópio Duarte Jr. (UFPR)
- Fabíola Gonçalves Pereira Greve (UFBA)
- Fatima Duarte Figueiredo (PUC-Minas)
- Fábio Luciano Verdi (UFSCar)
- Fernando Luís Dotti (PUC-RS)
- Flavia Coimbra Delicato (UFRJ)
- Francisco Vilar Brasileiro (UFCG)
- Geraldo Robson Mateus (UFMG)
- Gustavo Bittencourt Figueiredo (UFBA)
- Gustavo Sousa Pavani (UFABC)
- Heitor Soares Ramos Filho (UFAL)
- Horácio Antonio Braga Fernandes de Oliveira (UFAM)

- Humberto Torres Marques-Neto (PUC-Minas)
- Igor Monteiro Moraes (UFF)
- Ítalo Fernando Scotá Cunha (UFMG)
- Jacir Bordim (UnB)
- Jean-Marie Farines (UFSC)
- Jó Ueyama (USP)
- Joni da Silva Fraga (UFSC)
- José Augusto Suruagy Monteiro (UFPE)
- José Ferreira de Rezende (UFRJ)
- Jose Marcos Silva Nogueira (UFMG)
- José Neuman de Souza (UFC)
- Jussara M. Almeida (UFMG)
- Kelvin Lopes Dias (UFPE)
- Kleber Vieira Cardoso (UFG)
- Lau Cheuk Lung (UFSC)
- Leandro Aparecido Villas (UNICAMP)
- Leobino Nascimento Sampaio (UFBA)
- Liane Margarida Rockenbach Tarouco (UFRGS)
- Linnyer Beatrys Ruiz Aylon (UEM)
- Lisandro Zambenedetti Granville (UFRGS)
- Luci Pirmez (UFRJ)
- Lúcia Maria de Assumpção Drummond (UFF)
- Luciano Paschoal Gaspary (UFRGS)
- Luis C. E. De Bona (UFPR)
- Luis Henrique Costa (UFRJ)
- Luiz Eduardo Buzato (UNICAMP)
- Luiz Fernando Bittencourt (UNICAMP)
- Luiz Filipe Menezes Vieira (UFMG)
- Magnos Martinello (UFES)
- Marcel William Rocha da Silva (UFRRJ)
- Marcelo Dias de Amorim (LIP6 - França)
- Marcelo Gonçalves Rubinstein (UERJ)
- Marco Aurélio Spohn (UFFS)
- Marcos Augusto Menezes Vieira (UFMG)
- Markus Endler (PUC-Rio)
- Mauro Fonseca (UTFPR)
- Michael Stanton (RNP e UFF)
- Michele Nogueira (UFPR)
- Michelle Wangham (UNIVALI)
- Miguel Elias Mitre Campista (UFRJ)
- Moises Ribeiro (UFES)
- Nabor Mendonca (UNIFOR)
- Natalia Castro Fernandes (UFF)
- Nazareno Andrade (UFCG)
- Nelson L. S. da Fonseca (UNICAMP)
- Noemi Rodriguez (PUC-Rio)
- Olga Goussevskaia (UFMG)
- Otto Carlos Muniz Bandeira Duarte (UFRJ)
- Paulo Cunha (UFPE)

- Paulo de Figueiredo Pires (UFRJ)
- Paulo Henrique de Aguiar Rodrigues (UFRJ)
- Paulo André da Silva Gonçalves (UFPE)
- Pedro Braconnot Velloso (UFRJ)
- Pedro O. S. Vaz de Melo (UFMG)
- Rafael Pasquini (UFU)
- Rafael Pereira Esteves (IFRS)
- Raimundo José de Araújo Macêdo (UFBA e UFSB)
- Raquel Mini (PUC-Minas)
- Ricardo Augusto Rabelo Oliveira (UFOP)
- Rodolfo da Silva Villaça (IFES)
- Rodrigo Fonseca (BU - EUA)
- Ronaldo Alves Ferreira (UFMS)
- Ronaldo M. Salles (IME)
- Rosa M. M. Leão (UFRJ)
- Rossana Maria de Castro Andrade (UFC)
- Sidney Cunha de Lucena (UNIRIO)
- Silvana Rossetto (UFRJ)
- Stênio Flávio de Lacerda Fernandes (UFPE)
- Thais Vasconcelos Batista (UFRN)
- Wagner Meira Jr. (UFMG)
- Weverton Luis da Costa Cordeiro (UFRGS)
- William Ferreira Giozza (UnB)

Revisores da Trilha Principal

- Ademar Takeo Akabane (UNICAMP)
- Alberto Egon Schaeffer-Filho (UFRGS)
- Alberto Sampaio Lima (UFC)
- Aldelir Fernando Luiz (IFC)
- Aldri Luiz dos Santos (UFPR)
- Alex Borges Vieira (UFJF)
- Alextian Bartholomeu Liberato (IFES)
- Alfredo Goldman (USP)
- Allan Edgard Silva Freitas (IFBA)
- Allan Mariano de Souza (UNICAMP)
- Allan Vidal (UFSCar)
- Aloizio Silva (UFMG)
- Alyson de Jesus dos Santos (UFRJ)
- Alysson Neves Bessani (UL - Portugal)
- Américo Sampaio (UNIFOR)
- Ana Paula Couto da Silva (UFMG)
- André Castelo Branco Soares (UFPI)
- André Costa Drummond (UNB)
- Andre L. L. Aquino (UFAL)
- André Lage Freitas (UFAL)
- André Luiz Nasserala Pires (UFAC)
- Anelise Munaretto (UTFPR)
- Antonio Alfredo Ferreira Loureiro (UFMG)
- Antonio Augusto de Aragão Rocha (UFF)
- Antonio Carlos de Oliveira Jr. (UFG)
- Antonio Carlos de Oliveira Junior (UFG)
- Antônio Jorge Gomes Abelém (UFPA)
- Antonio Lobato (UFRJ)
- Antonio Tadeu Azevedo Gomes (LNCC)
- Artur Ziviani (LNCC)
- Atslands Rego da Rocha (UFC)
- Bruno Costa (IFRJ)
- Bruno Silvestre (UFG)
- Carlos Alberto Kamienski (UFABC)
- Carlos Alberto Vieira Campos (UNIRIO)
- Carlos André Guimarães Ferraz (UFPE)
- Carlos Maurício Seródio Figueiredo (UEA)
- Carlos Roberto Senna (UNICAMP)
- Celio Trois (UFPR)
- Célio Vinicius Neves de Albuquerque (UFF)
- Cesar Augusto Cavalheiro Marcondes (UFSCar)
- Christian Esteve Rothenberg (UNICAMP)
- Cintia Borges Margi (USP)
- Claudio Miceli de Farias (UFRJ)
- Cristiano Bonato Both (UFCSPA)
- Cristina Klippel Dominicini (IFES)
- Dalbert Matos Mascarenhas (CEFET-RJ)
- Daniel Cason (UFBA)

- Daniel de Oliveira (UFF)
- Daniel Fernandes Macedo (UFMG)
- Daniel Guidoni (UFSJ)
- Daniel Macêdo Batista (USP)
- Daniel Presser (UFSC)
- Daniel Sadoc Menasche (UFRJ)
- Danielo G. Gomes (UFC)
- Davi da Silva Böger (UFSC)
- Dênio Mariz Timóteo de Sousa (IFPB)
- Denis Rosário (UFPA)
- Dianne Scherly Varela de Medeiros (UFRJ)
- Diego Rossi Mafioletti (IFES)
- Diogo Mattos (UFRJ)
- Djamel Fawzi Hadj Sadok (UFPE)
- Dorgival Guedes (UFMG)
- Eder John Scheid (UFRGS)
- Edmundo de Souza e Silva (UFRJ)
- Edmundo Roberto Mauro Madeira (UNICAMP)
- Eduardo Cerqueira (UFPA)
- Eduardo Freire Nakamura (UFAM)
- Elias Procópio Duarte Jr. (UFPR)
- Elmano Ramalho Cavalcanti (IFPE)
- Erick Aguiar Donato (UNICAMP)
- Erik de Britto e Silva (UFMG)
- Evandro O. T. Salles (UFES)
- Fábio Luciano Verdi (UFSCar)
- Fabio Vieira (UFRJ)
- Fabíola Gonçalves Pereira Greve (UFBA)
- Fabrício Aguiar Silva (UFV)
- Fabricio Murai (UM-Amherst, EUA)
- Fátima Duarte Figueiredo (PUC-Minas)
- Fernanda Sumika Hojo de Souza (UFSJ)
- Fernando Augusto Teixeira (UFSJ)
- Fernando Luís Dotti (PUCRS)
- Flavia Coimbra Delicato (UFRJ)
- Flávio R. C. Sousa (UFC)
- Francisco Vilar Brasileiro (UFCG)
- Geraldo Robson Mateus (UFMG)
- Gilvan Martins Durães (IFBaiano)
- Gustavo Bittencourt Figueiredo (UFBA)
- Gustavo Maciel Dias Vieira (UFSCar)
- Gustavo Sousa Pavani (UFABC)
- Heitor Soares Ramos Filho (UFAL)
- Horácio Antonio Braga Fernandes de Oliveira (UFAM)
- Humberto Torres Marques-Neto (PUC-Minas)
- Hylson Vescovi Netto (UFSC)
- Ian Vilar Bastos (UFF)
- Igor Leão dos Santos (UFRJ)
- Igor Monteiro Moraes (UFF)

- Iguatemi E. Fonseca (UFPB)
- Islene Calciolari Garcia (UNICAMP)
- Ítalo Fernando Scotá Cunha (UFMG)
- Jacir Bordim (UNB)
- Jean-Marie Farines (UFSC)
- Jesús Martin Talavera Portocarrero (CEA-IoT)
- Jó Ueyama (USP)
- João Batista Pinto Neto (UFRO)
- João Guilherme Maia de Menezes (UFMG)
- João Vitor Torres (UFRJ)
- Joni da Silva Fraga (UFSC)
- Jorge Lima de Oliveira Filho (UESC)
- José Augusto Suruagy Monteiro (UFPE)
- José Ferreira de Rezende (UFRJ)
- Jose Marcos Silva Nogueira (UFMG)
- José Neuman de Souza (UFC)
- Joubert de Castro Lima (UFOP)
- Juliano Fischer Naves (IFRO)
- Julio Cesar Duarte (IME)
- Jussara M. Almeida (UFMG)
- Kelvin Lopes Dias (UFPE)
- Kleber Vieira Cardoso (UFG)
- Leandro Aparecido Villas (UNICAMP)
- Leandro Melo de Sales (UFAL)
- Leandro Nelinho Balico (UFRR)
- Leobino Nascimento Sampaio (UFBA)
- Leonardo Richter Bays (UFRGS)
- Leopoldo Mauricio (UFRJ)
- Liane Margarida Rockenbach Tarouco (UFRGS)
- Linnyer Beatrys Ruiz Aylon (UEM)
- Lucas Bondan (UFRGS)
- Lucas Muller (UFRGS)
- Luci Pirmez (UFRJ)
- Luciana Moreira S. de Souza (UFSC)
- Luciano Barreto (UFSC)
- Luciano Jerez Chaves (UNICAMP)
- Luciano Paschoal Gaspary (UFRGS)
- Luis C. E. De Bona (UFPR)
- Luís Henrique Costa (UFRJ)
- Luiz Eduardo Buzato (UNICAMP)
- Luiz Fernando Bittencourt (UNICAMP)
- Luiz Filipe Menezes Vieira (UFMG)
- Magnos Martinello (UFES)
- Maicon Kist (UFRGS)
- Maicon Melo Alves (UFF)
- Marcel William Rocha da Silva (UFRRJ)
- Marcelo Caggiani Luizelli (UFRGS)
- Marcelo Dias de Amorim (LIP6 - França)
- Marcelo Gonçalves Rubinstein (UERJ)

- Marcelo Menezes de Carvalho (UnB)
- Marcelo Pitanga Alves (UFRJ)
- Marcia Helena Moreira Paiva (UFES)
- Marco Aurélio Spohn (UFFS)
- Marcos Augusto Menezes Vieira (UFMG)
- Markus Endler (PUC-Rio)
- Martin Andreoni Lopez (UFRJ)
- Matias Artur Klafke Schimuneck (UFRGS)
- Mauro Fonseca (UTFPR)
- Maxwell E. Monteiro (IFES)
- Michael Stanton (RNP)
- Michele Nogueira (UFPR)
- Michelle Silva Wangham (UNIVALI)
- Miguel Cardoso Neves (UFRGS)
- Miguel Elias Mitre Campista (UFRJ)
- Moises Ribeiro (UFES)
- Muriel Figueredo Franco (UFRGS)
- Nabor Mendonca (UNIFOR)
- Natalia Castro Fernandes (UFF)
- Nazareno Andrade (UFCG)
- Nelson L. S. da Fonseca (UNICAMP)
- Nelson Rosa (UFPE)
- Noemi Rodriguez (PUC-Rio)
- Olga Goussevskaia (UFMG)
- Paulo André da Silva Gonçalves (UFPE)
- Paulo Cunha (UFPE)
- Paulo de Figueiredo Pires (UFRJ)
- Paulo Henrique de Aguiar Rodrigues (UFRJ)
- Paulo Henrique Mendes Maia (UECE)
- Paulo Manuel Mafra (SENAI)
- Paulo Romero Martins Maciel (UFPE)
- Paulo Sérgio Sausen (UNIJUÍ)
- Pedro Braconnot Velloso (UFRJ)
- Pedro de Botelho Marcos (FURG)
- Pedro O. S. Vaz de Melo (UFMG)
- Rafael Obelheiro (UDESC)
- Rafael Pasquini (UFU)
- Rafael Pereira Esteves (IFRS)
- Rafael Silva Guimarães (IFES)
- Rafaelli de Carvalho Coutinho (CEFET-RJ)
- Raimundo José de Araújo Macêdo (UFBA e UFSB)
- Raquel da S. Cabral (UFAL)
- Raquel Mini (PUC-Minas)
- Reinaldo Cézar de Morais Gomes (UFCG)
- Ricardo Augusto Rabelo Oliveira (UFOP)
- Ricardo Pfitscher (UFRGS)
- Roberto Iraja Tavares da Costa Filho (UFRGS)
- Roberto Sadao Yokoyama (UTFPR)
- Rodolfo Antunes (UFRGS)

- Rodolfo da Silva Villaça (IFES)
- Rodrigo de Souza Couto (UERJ)
- Rodrigo Fonseca (BU - EUA)
- Rodrigo Stange Tessinari (UFES)
- Ronaldo Alves Ferreira (UFMS)
- Ronaldo M. Salles (IME)
- Rosa M. M. Leão (UFRJ)
- Rossana Maria de Castro Andrade (UFC)
- Sand Luz Correa (UFG)
- Sergio Armando Gutiérrez (UNAL)
- Sidney Cunha de Lucena (UNIRIO)
- Silvana Rossetto (UFRJ)
- Silvia Cercos (DTU, Dinamarca)
- Stênio Flávio de Lacerda Fernandes (UFPE)
- Taniro Chacon Rodrigues (UFRN)
- Thais Vasconcelos Batista (UFRN)
- Thiago Bruno Melo de Sales (UFAL)
- Ubiratam Carvalho de Paula Junior (UFRRJ)
- Uéverton dos Santos Souza (UFF)
- Vinicius da Cunha M. Borges (UFG)
- Vinícius Fernandes Soares Mota (UFMG)
- Vinicius Fonseca e Silva (UFMG)
- Vinícius Morais (UFMG)
- Vinícius Tavares Guimarães (IFSul)
- Wagner Meira Jr. (UFMG)
- Wendley Souza da Silva (UFC)
- Weverton Luis da Costa Cordeiro (UFRGS)
- William Ferreira Giozza (UnB)
- Yeda Regina Venturini (UFSCar)

Comitê de Programa do Salão de Ferramentas

- Alfredo Goldman (USP)
- Antonio Augusto de Aragão Rocha (UFF)
- Bruno Schulze (LNCC)
- Carlos Alberto Kamienski (UFABC)
- Carolina Felicíssimo (RNP)
- Cesar Augusto Cavalheiro Marcondes (UFSCar)
- Cintia Borges Margi (USP)
- Daniel Cordeiro (USP)
- Divanilson Campelo (UFPE)
- Eduardo Cerqueira (UFPA)
- Fábio Luciano Verdi (UFSCar)
- Flavia Coimbra Delicato (UFRJ)
- Flávio de Oliveira Silva (UFU)
- Gustavo Bittencourt Figueiredo (UFBA)
- Hélio Guardia (UFSCar)
- Hermes Senger (UFSCar)
- Kalinka Castelo Branco (USP)
- Leandro Aparecido Villas (UNICAMP)
- Leobino Nascimento Sampaio (UFBA)
- Lisandro Zambenedetti Granville (UFRGS)
- Luiz Claudio Theodoro (Algar Telecom)
- Luiz Fernando Bittencourt (UNICAMP)
- Marcelo Barcelos (DATACOM)
- Marcos Salvador (Lenovo)
- Nelson Rosa (UFPE)
- Rafael Scaraficci (CPqD)
- Raphael Camargo (UFABC)
- Rodolfo da Silva Villaça (IFES)
- Ronaldo Alves Ferreira (UFMS)
- Yeda Regina Venturini (UFSCar)

Revisores do Salão de Ferramentas

- Alfredo Goldman (USP)
- André Ricardo Abed Grégio (CTI/MCTI)
- Antonio Augusto de Aragão Rocha (UFF)
- Carlos Alberto Kamienski (UFABC)
- Cesar Augusto Cavalheiro Marcondes (UFSCar)
- Cintia Borges Margi (USP)
- Daniel Cordeiro (USP)
- Daniel Massami Muniz Yokoyama (LNCC)
- Danilo Carastan dos Santos (UFABC)
- Denis Rosário (UFPA)
- Divanilson Campelo (UFPE)
- Eduardo Cerqueira (UFPA)
- Erik de Britto e Silva (UFMG)
- Fábio Luciano Verdi (UFSCar)
- Flavia Coimbra Delicato (UFRJ)
- Flávio de Oliveira Silva (UFU)
- Frances Santos (UNICAMP)
- Gustavo Bittencourt Figueiredo (UFBA)
- Hélio Guardia (UFSCar)
- Henrique Araujo (UFRJ)
- Henrique Klôh (UFF)
- Hermes Senger (UFSCar)
- Kalinka Castelo Branco (USP)
- Kelvin Lopes Dias (UFPE)
- Leobino Nascimento Sampaio (UFBA)
- Luis Sant Ana (UFABC)
- Luiz Claudio Theodoro (Algar Telecom)
- Luiz Arthur Feitosa dos Santos (UTFPR)
- Luiz Fernando Bittencourt (UNICAMP)
- Luiz Henrique Andrade Correia (UFLA)
- Marcos Salvador (Lenovo)
- Martin Andreoni Lopez (UFRJ)
- Nelson Rosa (UFPE)
- Roberto Sadao Yokoyama (UTFPR)
- Rodolfo da Silva Villaça (IFES)
- Ronaldo Alves Ferreira (UFMS)

Sumário

Trilha Principal do SBRC 2016

Trilha Principal do SBRC 2016
Sessão Técnica 1
Caracterização de Comportamento

Trabalho Colaborativo em Serviços de Armazenamento na Nuvem: Uma Análise do Dropbox*

Glauber Dias Gonçalves[1], Alex Borges Vieira[2], Ana Paula Couto da Silva[1], Jussara M. Almeida[1]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais

[2]Departamento de Ciência da Computação - Universidade Federal de Juiz de Fora

{ggoncalves, ana.coutosilva, jussara}@dcc.ufmg.br

alex.borges@ufjf.edu.br

Resumo. *Provedores de serviços de armazenamento na nuvem adotam várias medidas para incentivar a atividade de seus usuários bem como atrair novos usuários. Uma das ações adotadas pelos principais provedores com esse propósito é oferecer a opção de realizar trabalho colaborativo, via compartilhamento de dados entre usuários em rede. Neste estudo, nós investigamos a relação entre trabalho colaborativo e o nível de atividade dos usuários do Dropbox, um dos serviços mais populares atualmente. Nós medimos o volume do tráfego que os usuários geram à medida que armazenam dados na nuvem e realizam compartilhamentos, utilizando dados coletados de quatro redes distintas. Nossos resultados indicam que usuários engajados em trabalhos colaborativos sincronizam maior volume de dados com a nuvem. Logo, esses usuários tem maior atividade e potencialmente pagam ou podem vir a pagar pelo serviço, sugerindo que, apesar dos custos, a funcionalidade de trabalho colaborativo pode aumentar a receita dos provedores de armazenamento na nuvem.*

Abstract. *Cloud storage service providers have adopted several measures to incentivize the activity of their users as well as attract new ones. One of actions adopted by the major providers for this purpose is to provide the option to carry out collaborative work via data sharing between networked users. In this study, we investigate the relationship between collaborative work and the activity level of the users of Dropbox, one of the most popular services currently. We measure the traffic volume that users generate as they store data in the cloud and make shared folders using data collected from four different networks. Our results indicate that users participating in collaborative work synchronize more data to the cloud. Therefore, these users have higher activity and potentially can pay for the service, suggesting that, despite its cost, collaborative work feature can increase revenue of cloud storage providers.*

1. Introdução

O uso ubíquo e transparente de serviços de armazenamento em nuvem foi determinante para impulsionar sua popularização. É notório o crescente uso de aplicações desse tipo,

*Esta pesquisa é financiada pelo projeto FAPEMIG-PRONEX-MASWeb – Modelos, Algoritmos e Sistemas para a Web (processo APQ-01400-14), pelo INCTWeb (MCT/CNPq 573871/2008-6) e CNPq.

desde usuários domésticos a grandes empresas. O mercado é dominado por grandes empresas como Dropbox, Microsoft, Amazon e Apple que atualmente movimentam, apenas nos EUA, valores acima de $12 bilhões de dólares [Gracia-Tinedo et al. 2015].

Em tais serviços, o compartilhamento de dados – crucial para trabalho colaborativo – é uma das funcionalidades oferecidas. Recentemente, tem se observado um empenho dos maiores provedores de armazenamento na nuvem em aumentar colaborações entre usuários em seus serviços. Por exemplo, Dropbox recentemente lançou a versão empresarial de seu serviço com foco em compartilhamentos, enquanto Google Drive busca incentivar o trabalho colaborativo via edição de texto, planilhas eletrônicas e apresentações diretamente na nuvem em tempo real. De fato, uma pesquisa realizada em ambiente coorporativo [Shami et al. 2011] mostra que compartilhamento de dados permite o aumento de coleções de recursos digitais disponíveis em empresas, como consequência, a utilização de sistemas de arquivos torna-se maior.

Em tese, quanto maior a atividade de um usuário, mais dados ele irá armazenar e, consequentemente, maior será a receita dos serviços de armazenamento na nuvem. Geralmente, os provedores desses serviços incentivam a atividade de seus usuários por ações de marketing ou por promoções temporárias. Como alternativa a essas medidas, trabalho colaborativo via compartilhamento de arquivos pode atrair novos usuários e até mesmo incentivar a atividade dos já existentes. Intuitivamente, ao se compartilhar dados no serviço de armazenamento, todos os envolvidos são onerados pelo espaço em disco compartilhado (a custo de um único recurso na nuvem). Mais ainda, novos usuários podem se juntar ao serviço para participar de um projeto ou acessar um arquivo compartilhado.

Compartilhamento de dados parece ser uma característica chave que atrai os usuários para a serviços de armazenamento na nuvem. No entanto, essa característica gera custos para o serviço. Tomando Dropbox como um exemplo, a menos que os usuários estejam localizados em uma única rede[1], todos os participantes em uma pasta compartilhada tem que recuperar dados a partir da nuvem. Isso potencialmente cria uma carga de trabalho extra para servidores, além de gerar custos com largura de banda [Gonçalves et al. 2015].

Dado a importância e os custos de trabalhos colaborativos via compartilhamento de dados para os serviços de armazenamento na nuvem, neste trabalho buscamos entender até que ponto essa funcionalidade pode revelar o perfil de atividade do usuário. Nesse sentido, nós buscamos responder as seguintes questões: *(Q1) A atividade de um usuário em serviços de armazenamento na nuvem pode ser medida pela sua quantidade de colaborações? (Q2) Qual o impacto de trabalho colaborativo na utilização de serviços de armazenamento na nuvem em grandes redes?* Note que, embora alguns estudos tenham abordado características de compartilhamentos em serviços de armazenamento na nuvem, essas questões ainda não foram respondidas, visto que o foco desses estudos eram padrões de carga e tráfego [Gonçalves et al. 2014] e aspectos de arquitetura visando melhorar o suporte a compartilhamento de dados [Gonçalves et al. 2015].

Para tal, nós conduzimos análises detalhadas sobre essas questões no Dropbox — um dos serviços mais populares atualmente. Todas as nossas análises são baseadas em dados de tráfego coletados em quatro redes distintas: duas redes universitárias (uma na América do Sul e outra na Europa) e dois Pontos de Presença (PoP) de um provedor de

[1]Dropbox implementa o protocolo `LanSync` para sincronização local (fora do escopo desse trabalho).

serviço de Internet (ISP) europeu[2]. Mais especificamente, nós correlacionamos o nível de atividade dos usuários, em termos de volume de dados sincronizados com a nuvem, com características do Dropbox relacionadas a trabalho colaborativo.

Nossos resultados indicam que usuários engajados em trabalhos colaborativos sincronizam maior volume de dados com a nuvem. Logo, esses usuários tem maior atividade e potencialmente pagam ou podem vir a pagar pelo serviço. Visto que trabalho colaborativo gera custos para o provedor, nosso estudo é importante porque aponta justificativas para o provedor investir em tal funcionalidade, que pode levar a um aumento de receita, assim como atrair novos usuários. Em suma, nossos principais resultados são: (i) o número de compartilhamentos do usuário indica o seu nível de atividade, em volume de sincronizações de dados com a nuvem, dado que existe uma correlação alta e não linear entre essas duas variáveis em diferentes redes; e (ii) três perfis de atividade podem descrever o comportamento dos usuários de serviços de armazenamento na nuvem em redes de universidades e ISPs: usuários que focam em colaborações dentro da rede ou colaborações externas à rede, e usuários que focam basicamente em armazenar dados.

A seguir, na Seção 2 são discutidos trabalhos relacionados. Na Seção 3 são apresentados conceitos básicos do Dropbox e a metodologia adotada para coleta e análise de dados. Na Seção 4 são discutidos resultados da correlação entre o nível de atividade dos usuários e características do Dropbox relacionadas a trabalho colaborativo. Impactos de trabalhos colaborativos nas redes monitoradas e a identificação/caracterização de perfis de usuários são tratados na Seção 5. Por fim, na Seção 6 conclui-se esse estudo.

2. Trabalhos Relacionados

Este trabalho analisa a relação de trabalhos colaborativos via compartilhamento de dados com a atividade dos usuários em sistema de armazenamento na nuvem por meio de medições de tráfego em várias redes. Nosso trabalho mais recente [Gonçalves et al. 2015] também analisa compartilhamentos no Dropbox, mas com foco específico em downloads redundantes e seu impacto no tráfego de uma rede universitária. Em [Costa et al. 2014] foi avaliado alguns aspectos do compartilhamento entre usuários do Dropbox usando uma coleta de dados de voluntários. Diferente desses, este trabalho faz uma análise mais abrangente sobre a utilização de recursos de compartilhamentos do Dropbox e o seu impacto na atividade do usuário e no tráfego de dados do serviço, considerando redes acadêmicas e residenciais.

Serviços de armazenamento na nuvem têm recebido grande atenção da comunidade científica. Em [Drago et al. 2012] é apresentado a primeira caracterização detalhada acerca da utilização e desempenho do Dropbox. No presente trabalho, nós contamos, em parte, com a metodologia de coleta e processamento de dados desenvolvida em [Drago et al. 2012]. Porém, nós estendemos essa metodologia para uma nova análise sobre o volume de dados sincronizado por usuários do Dropbox. Essa análise utiliza a combinação entre o volume dos fluxos de dados e as mensagens de sincronização dos clientes Dropbox. Tal método foi preliminarmente desenvolvido em nosso trabalho anterior [Gonçalves et al. 2015].

Grande parte dos estudos sobre serviços de armazenamento na nuvem focam em aferição (*benchmarking*). Nesse sentido, esses trabalhos utilizam primordialmente

[2]Os leitores interessados podem contactar os autores para acessar o conjunto de dados.

medições ativas, que são experimentos realizados em ambiente controlado para avaliar alguma funcionalidade do serviço. Em [Wang et al. 2012] são analisados gargalos do Dropbox, enquanto APIs de três provedores desse tipo de serviço são avaliadas em [Gracia-Tinedo et al. 2013]. Um método para medir a eficiência do uso de tráfego em serviços de armazenamento na nuvem é apresentado em [Li et al. 2014], enquanto um *framework* para realizar aferições automáticas de vários aspectos de aplicações clientes foi proposto em [Bocchi et al. 2015a].

Outros trabalhos propõem novos protocolos e arquiteturas para melhorar o desempenho, ou reduzir custos de serviços de armazenamento nas nuvens. Em [Chen et al. 2014] foi proposto um *framework* que coleta informações semânticas sobre os arquivos (e.g., tipo de arquivo) visando melhorar o desempenho do serviço. Um mecanismo que garante consistência entre os arquivos locais (em vários dispositivos que o compartilham) e o repositório remoto foi apresentado em [Zhang et al. 2014]. Em [Cui et al. 2015] foi proposto QuickSync, um sistema que otimiza a sincronização entre dispositivos e a nuvem combinando técnicas como *network-aware chunking*. Nosso trabalho contribui diretamente para esses esforços uma vez que, cientes da relação entre trabalhos colaborativos e a atividade dos usuários, os desenvolvedores podem propor protocolos e arquiteturas otimizados para esses serviços.

A Qualidade de Experiência (QoE) do usuário em serviços de armazenamento na nuvem foi estudada em [Amrehn et al. 2013, Casas and Schatz 2014]. Os autores realizaram experimentos para correlacionar QoE e medições de QoS, observando que largura de banda é a métrica de QoS mais sensível para os usuários em termos de QoE. Essas correlações são importantes para orientar ISPs, provedores de armazenamento na nuvem e usuários sobre os requisitos de largura de banda necessárias para uma utilização satisfatória do serviço. Por outro lado, o nosso objetivo é ir além desses requisitos técnicos básicos, medindo funcionalidades do serviço como trabalho colaborativo que podem ter impacto significativo sobre a atividade do usuário.

Finalmente, estudos mais recentes caracterizaram o comportamento de usuários a partir de dados coletados em redes ou provedores de serviço. Em [Gracia-Tinedo et al. 2015] foi apresentado um estudo de medição do serviço Ubuntu One (U1). Os autores observaram que U1 é usado mais como um serviço de armazenamento do que trabalho colaborativo, dado que apenas 1,8% dos usuários compartilham dados nesse serviço. Um estudo comparativo sobre a utilização de armazenamento na nuvem em diferentes terminais foi proposto em [Bocchi et al. 2015b]. Os autores constataram que o volume de *uploads* é maior em terminais móveis, possivelmente associado ao armazenamento de conteúdo multimedia, ao passo que o volume de *downloads* é maior em PCs devido à sincronização desse conteúdo. Nenhum desses estudos investigam trabalho colaborativo e a atividade dos usuários, como proposto no presente estudo.

3. Conceitos e Metodologia de Coleta de Dados

Antes da apresentação de nossas análises, nós sumarizamos o mecanismo básico de sincronização de dados no Dropbox. Informações mais detalhadas e uma descrição abrangente dos mecanismos internos do Dropbox podem ser encontradas em [Drago et al. 2012, Gonçalves et al. 2016].

3.1. Dropbox: Mecanismo Básico

O Dropbox sincroniza dados utilizando dois conceitos principais: *dispositivos* e *namespaces*. Os usuários podem registrar vários dispositivos nesse serviço. Durante esse processo, os usuários devem selecionar uma pasta inicial a partir de onde os arquivos são sincronizados com a nuvem. Esta pasta inicial é visível em qualquer outro dispositivo pertencente ao usuário. Os usuários podem também compartilhar dado com outros usuários através da criação de pastas compartilhadas, que são visíveis em todos os dispositivos de todos os usuários que participam do compartilhamento. A pasta inicial e as pastas compartilhadas são raízes de árvores de diretórios independentes conhecidas como *namespaces* no Dropbox[3].

Nós nos concentramos apenas na aplicação cliente para PC, uma vez que ela é responsável por mais de 75% do tráfego Dropbox [Bocchi et al. 2015b]. Os dispositivos que utilizam esse cliente mantêm, geralmente, uma cópia local de todos os arquivos presentes nos *namespaces* do usuário. A adição de qualquer dado em um *namespace* desencadeia a propagação desse dado: todos os dispositivos com o cliente, e com o *namespace* registrado, recuperam o dado imediatamente, caso estejam em execução (ou tão logo eles iniciem sua execução).

Internamente, Dropbox controla o estado dos *namespaces* por meio de um *protocolo de notificações*. Em resumo, cada *namespace* está associado a um identificador monotônico (i.e., *Journal Identifier* ou JID), representando sua atualização mais recente. Dispositivos descobrem se há atualizações pendentes para *namespaces* via troca periódica de uma lista de *namespaces* e seus respectivos JIDs com os servidores do Dropbox. Se algum *namespace* está desatualizado em um dispositivo, esse dispositivo executa várias transações com os servidores Dropbox até que todos os *namespaces* estejam sincronizados com a nuvem.

Assim, pela observação das mensagens do protocolo de notificação, é possível identificar quando os *namespaces* são atualizados. De fato, nós desenvolvemos um método em [Gonçalves et al. 2016] para coletar dados a longo prazo de um conjunto de *namespaces* do Dropbox em grandes redes como campi universitários e ISPs. O nosso método inclui estimativas para o volume de tráfego transferido em cada transição de JIDs em uma grande amostragem de *namespaces*. No presente trabalho nós exploramos o conjunto de dados obtidos por esse método para analisar trabalhos colaborativos no Dropbox.

3.2. Coleta de Dados

Nós contamos com dados capturados e pré-processados em nosso trabalho anterior [Gonçalves et al. 2016] para realizar o estudo atual. Esses dados foram capturados através da monitoração do tráfego Dropbox em 4 pontos de coleta diferentes, incluindo dois campi universitários e duas redes de ISPs.

Mais precisamente, os pontos de coleta Campus-1 e Campus-2 estão em redes de campus distintos, na América do Sul e Europa, respectivamente. Campus-1 tem uma população de usuários (incluindo professores, alunos e funcionários) de aproximadamente 57.000 pessoas, enquanto o Campus-2 serve cerca de 15.000 pessoas. PoP-1 e PoP-2 monitoram clientes em dois pontos de presença (PoPs) de um ISP europeu, agregando mais de 25.000 e 5.000 residências, respectivamente.

[3] `https://blogs.dropbox.com/tech/2014/07/streaming-file-synchronization`

Nossos dados incluem (i) informações sobre o volume de tráfego trocado por clientes com servidores Dropbox; (ii) metadados extraídos de mensagens de notificação Dropbox. Esse último foi capturado por meio de uma inspeção detalhada de pacotes e inclui, para cada mensagem, um identificador do dispositivo cliente e sua lista de *namespaces* com respectivos JIDs. Note que as mensagens de notificação não oferecem informações sobre identidades de usuários. Mais ainda, por razões de privacidade, os endereços IP dos clientes são anonimizados nos pontos de coleta de dados. Analisamos 27 k dispositivos Dropbox únicos e 61 k *namespaces* exclusivos durante cerca de 1 ano de coleta de dados. No total, coletamos mais de 20 TB de tráfego trocados com servidores Dropbox, onde observamos que o volume de compartilhamento de dados é significativo: mais de 60% do tráfego coletado é *download* e até 70% dos *downloads*, dependendo da rede[4], estão relacionados com pastas compartilhadas.

Os clientes Dropbox, quando em execução, trocam mensagens de notificação com os servidores a cada minuto aproximadamente. Portanto, nós rastreamos essas notificações para reconstruir um conjunto de dados que resume as atualizações realizadas nos *namespaces*. Nós estimamos o volume de cada atualização, correlacionando o volume do fluxo de dados de cada cliente com as suas mensagens de notificação: considerando os protocolos Dropbox [Drago et al. 2012], notificações a respeito de uma nova atualização de *namespaces* são observadas na rede próximas (antes/depois) à transmissão do fluxo de dados. Assim, ao combinar esses fluxos com as mensagens de notificação (quase) simultâneas de um cliente, nós associamos mais de 63% do tráfego Dropbox com mensagens de notificação. Essa heurística mostrou-se capaz de produzir uma boa estimativa para o volume de tráfego de cada atualização em *namespaces* Dropbox como descrevemos com mais detalhes em [Gonçalves et al. 2016].

4. Análise da Atividade de Usuários

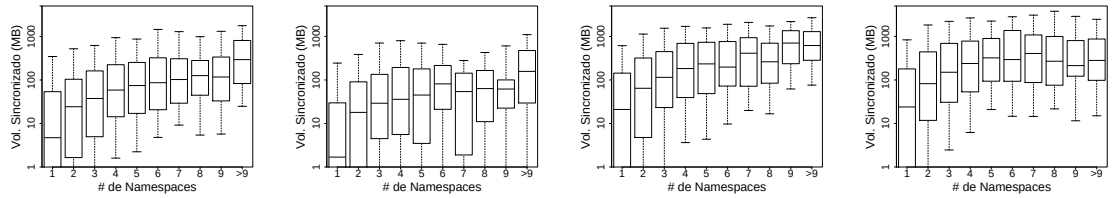
Nesta seção analisamos como métricas associadas ao trabalho colaborativo se correlacionam com atividade dos usuários no Dropbox. Nosso objetivo é responder a questão *Q1: A atividade de um usuário em serviços de armazenamento na nuvem pode ser medida pela sua quantidade de colaborações?*

Nesse sentido, definimos como atividade do usuário o volume de suas sincronizações ao longo de um período. Tais sincronizações correspondem aos *uploads* e *downloads* entre o(s) dispositivo(s) do usuário e os servidores remotos de armazenamento do Dropbox.

Entre os principais recursos que usuários utilizam no serviço de armazenamento na nuvem para trabalho colaborativo, nós analisamos a correlação entre (1) número de *namespaces*; (2) número de dispositivos e; (3) número de parcerias, i.e., demais usuários da rede ligados às pastas compartilhadas do usuário sendo analisado; com a atividade desse usuário. O volume total de atividades dos usuários, bem como dessas três métricas foram agrupados em intervalos de tempo de um mês. Desse modo, foi possível considerar usuários com diferentes frequências de utilização do serviço. Adicionalmente, um usuário pode ter mais de uma amostra referente a sua atividade no nosso conjunto de dados.

As correlações foram calculadas da seguinte forma: primeiro, as métricas de trabalho colaborativo foram agrupadas de acordo com o seu valor. Por exemplo, ao corre-

[4]Por restrição de espaço, nós detalhamos em [Gonçalves et al. 2016] as informações de cada rede.



(a) Campus-1 (b) Campus-2 (c) PoP-1 (d) PoP-2

Figura 1. Número de *namespaces* vs. Volume de Sincronizações dos Usuários.

Tabela 1. Coeficiente de correlação linear (Pearson) e não linear (Spearman) entre atividade dos usuários e métricas relacionadas a trabalhos colaborativos.

	Correlaçao Linear (Pearson)				Correlaçao Nao linear (Spearman)			
	Campus-1	Campus-2	PoP-1	PoP-2	Campus-1	Campus-2	PoP-1	PoP-2
Parcerias	0,44	0,65	0,83	-0,42	0,26	0,59	0,79	-0,48
Dispositivos	0,31	0,30	0,49	0,61	0,35	0,68	0,48	0,64
Compartilhamentos	0,87	0,84	0,90	0,69	0,99	0,92	0,96	0,59

lacionar quantidade de *namespaces* do usuário com a sua atividade, nós criamos grupos de tamanho 1, 2, 3, ou mais *namespaces*. Acumulamos usuários com valores acima de 9 *namespaces* em um único grupo, dado que observamos menos usuários com tais valores. Por fim, calculamos a correlação entre a métrica e o valor da mediana da distribuição da atividade dos usuários em cada grupo.

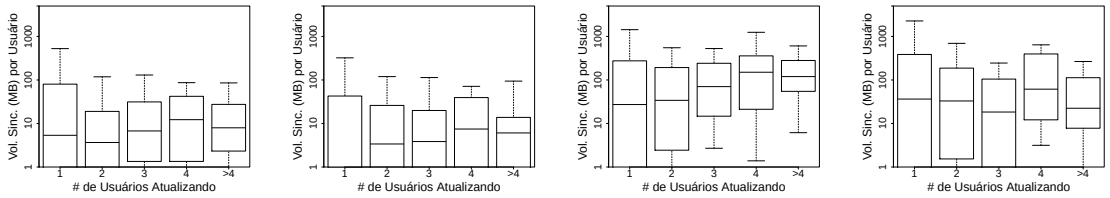
A Figura 1 exemplifica as correlações calculadas, com a métrica total de *namespaces* por mês, que apresentou maior correlação com a atividade do usuário (Tabela 1). O processo para calcular as correlações é análogo para as demais métricas, e seus gráficos foram omitidos por questão de espaço. Nessa figura, para cada total de *namespaces*, usamos o *box-plot* para sumarizar a distribuição da atividade dos usuários: o retângulo central se expande entre o primeiro e terceiro quartil, o segmento interior é a mediana, enquanto os indicadores abaixo e acima do retângulo representam o 9^{o} e 91^{o} percentis.

Note que a distribuição da atividade em cada grupo é muito variável e tende a valores pequenos, visto pela concentração de volumes entre 9^{o} percentil e o 3^{o} quartil (66% dos dados). Logo, consideramos a mediana como o valor mais adequado para representar a atividade por grupo. Observe na Figura 1 que a mediana da atividade dos usuários tende a crescer com o aumento do número de *namespaces*. Relembre (Seção 3.1) que cada *namespace* se refere à pasta inicial acessada apenas por um usuário Dropbox ou às demais pastas que são compartilhadas entre diferentes usuários do Dropbox. Logo, nossos resultados mostram que a intensidade de atividades dos usuários está relacionada à sua quantidade de trabalhos colaborativos (i.e., *namespaces*). Note que usuários que sincronizam mais que cinco *namespaces* por mês apresentam comportamento mais homogêneo, com uma distribuição mais concentrada em torno da mediana e tendem a sincronizar volumes altos de dados, i.e., mediana acima de 100MB e próxima a 1GB nas redes residenciais.

A Tabela 1 mostra os resultados dos coeficientes de correlação de Pearson[5] e de Spearman[6] [Jain 1991] referentes a cada uma das correlações calculadas. Note que o número de *namespaces* é a métrica que apresenta maior correlação com a atividade do

[5]Mede a correlação linear entre duas variáveis.

[6]Mede a correlação não-linear entre duas variáveis.



(a) Campus-1 (0.8) (b) Campus-2 (0.9) (c) PoP-1 (0.9) (d) PoP-2 (-0.2)

Figura 2. Número de usuários ativos em um *namespace* vs. volume médio de sincronizações por usuário, com respectivos coeficientes de correlação não linear (Spearman) para cada rede.

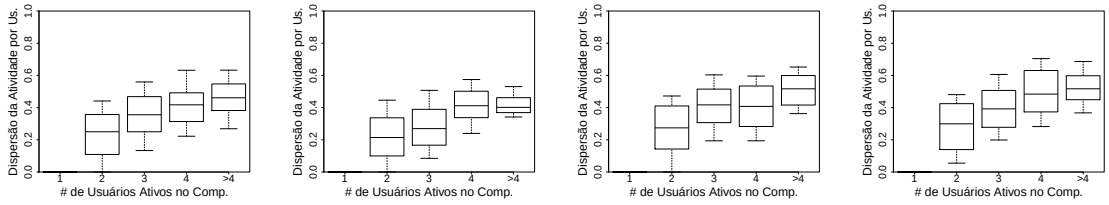
usuário em todas as redes. É importante observar que, essa correlação para as redes Campus-1, Campus-2 e PoP-1 tende a ser não linear (note que o coeficiente de Spearman é maior que o de Pearson). Isso indica que trabalhos colaborativos nessas redes estão realacionadas a uma atividade ainda mais expressiva dos usuários, i.e., volume maior de sincronizações.

Dado a importância do número de *namespaces* para caracterizar a atividade dos usuários, analisamos também as colaborações (i.e., compartilhamentos) que ocorrem somente internamente às redes analisadas. Para isso, nós calculamos o volume de sincronizações por *namespace* e dividimos esse volume igualmente pelo total de usuários ativos no *namespace*[7]. Assim, obtivemos a atividade média por usuário colaborando em um *namespace* dentro da rede.

A Figura 2 mostra a distribuição para o volume de sincronizações por usuário em *namespaces* dentro das redes. Mantivemos os *namespaces* com apenas um usuário ativo para efeito de comparação com multiplos usuários. Pode-se observar que existe uma correlação não linear alta (i.e., acima de 0,6) entre o volume de sincronizações mediano por usuário e o total de usuários ativos em *namespaces*. A correlação é baixa apenas no PoP-2 possivelmente devido a flutuações do volume de sincronizações de alguns *namespaces*, que foram maiores nessa rede. Contudo, quando consideramos o número de modificações por usuário ao invés do volume, observamos uma correlação acima de 0,65 em todas as redes. Esses resultados mostram indícios que os compartilhamentos dentro das redes tem um potencial para incentivar a atividade dos usuários.

Observamos ainda se poucos usuários dentro das redes são responsáveis pela maior parte do volume de dados sincronizados em cada *namespace*. Para isso nós calculamos o coeficiente de Gini como indicador de desigualdade [Cowell 2011]. Esse coeficiente varia entre 0 (completa igualdade) e 1 (apenas um usuário sincronizou dados no *namespace*). A Figura 3 mostra que a atividade por usuário se torna mais desequilibrada à medida que o *namespace* tem mais usuários. Em redes residenciais (PoPs) o coeficiente de Gini atinge valores superiores a 0,5 em *namespaces* com mais de 4 usuários. De fato, observamos que o usuário mais ativo impacta em pelo menos 40% (porcentagem mediana) da atividade total por *namespace* em todas as redes. Note que mesmo havendo esse desequilibrio, os dados do *namespace* provavelmente foram sincronizados para todos os seus usuários igualmente, estando alguns deles fora das redes monitoradas.

[7]Consideramos apenas os usuários registrados no *namespace* que realizaram alguma sincronização de dados dentro das redes monitoradas.



(a) Campus-1 (b) Campus-2 (c) PoP-1 (d) PoP-2

Figura 3. Dispersão da atividade do usuário medida entre 0-1 (Gini) vs. número de usuários ativos no *namespace*.

Resumindo, os resultados apresentados nessa seção mostram indícios de que o engajamento de usuários em trabalhos colaborativos, medidos pelo número de *namespaces* do usuário (Figura 1) ou sincronizações em compartilhamentos dentro das redes (Figura 2), podem revelar quais são os usuários mais ativos em sistemas de armazenamento em nuvem. Em [Drago et al. 2012, Gracia-Tinedo et al. 2015] esses usuários são apontados como *heavy users* pois são eles quem impactam o sistema de armazenamento em si. No entanto, apesar de definirem o que são *heavy users* tais estudos não observam a correlação existente entre trabalhos colaborativos e a atividade desses usuários.

5. Trabalho Colaborativo em Redes

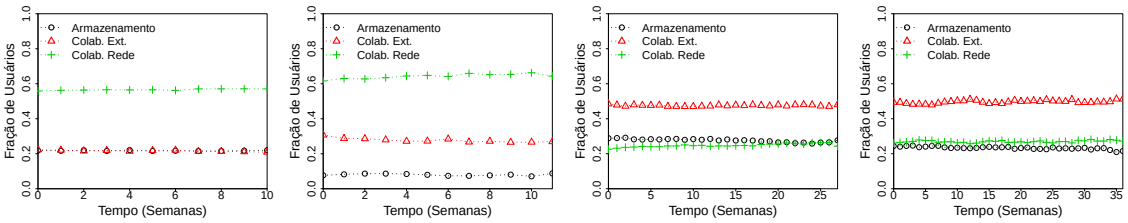
O foco principal desta seção é responder a questão *Q2: Qual o impacto de trabalhos colaborativos na utilização de serviços de armazenamento na nuvem em grandes redes?* Nesse sentido, propomos um modelo que considera diferentes estados de atividade dos usuários em redes acadêmicas e residenciais (Seção 5.1). A partir desse modelo analisamos os perfis de usuários baseados em transições entre esses estados e caracterizamos a carga desses perfis no serviço de armazenamento (Seção 5.2).

5.1. Estados de Atividade do Usuário

Modelamos as atividades dos usuários considerando três diferentes estados: (1) *Colaborações em Rede*: usuários colaborando com outros usuários internamente às redes monitoradas; (2) *Colaborações Externas*: usuários colaborando com usuários externos às redes monitoradas; e (3) *Armazenamento*: usuários cujo foco principal é basicamente armazenar dados na nuvem e sincronizá-los entre seus dispositivos.

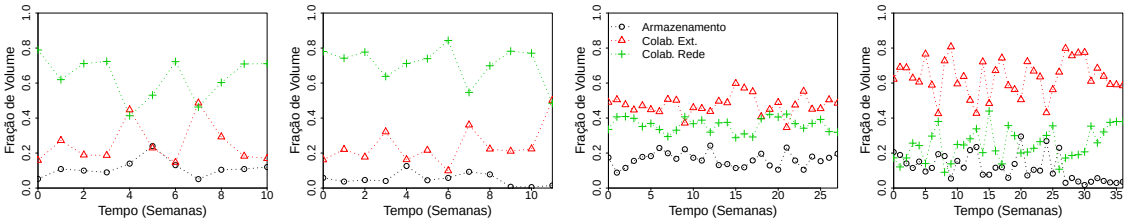
No primeiro estado de atividade (*Colaborações em Rede*), incluímos usuários que sincronizaram dados em ao menos um *namespace* compartilhado dentro da rede monitorada. Nos demais estados, incluímos usuários cujas sincronizações ocorreram apenas em *namespaces* não compartilhados dentro da rede. Esses *namespaces* podem ser a pasta inicial do usuário e/ou pastas compartilhadas com usuários externos à rede. Assim, o segundo estado (*Colaborações Externas*) contém usuários que sincronizam mais de um *namespace*, ou seja, usuários que realizaram colaborações com usuários externos à rede, dado que o Dropbox define uma pasta inicial por usuário (ver Seção 3.1). Por sua vez, o terceiro estado (*Armazenamento*) contém usuários que sincronizaram um único *namespace*. Como esse *namespace* pode se tratar da pasta inicial, conjecturamos que o foco desses usuários é armazenar dados na nuvem e sincronizá-los entre seus dispositivos.

A seguir, nós analisamos o número de usuários e o volume de sincronizações para cada estado de atividade por semana. A análise semanal evita os efeitos da sazo-



(a) Campus-1 (b) Campus-2 (c) PoP-1 (d) PoP-2

Figura 4. Fração de usuários nos três estados de atividade ao longo do tempo.



(a) Campus-1 (b) Campus-2 (c) PoP-1 (d) PoP-2

Figura 5. Fração do volume sincronizado nos três estados de atividade ao longo do tempo.

nalidade que ocorrem em diferentes dias, por exemplo, finais de semanas e dias úteis. Adicionalmente, analisamos apenas usuários estáveis, i.e., usuários monitorados em pelo menos metade do período de coleta em cada rede. Essa medida diminui a interferência de usuários novos ou visitantes às redes monitoradas nos resultados de nossas análises.

A Figura 4 mostra a fração de usuários em cada estado ao longo do tempo nas redes monitoradas. De uma forma geral, observa-se que colaborações em rede são dominantes nos campi (Campus-1 e Campus-2) e menos presentes nas residências (PoP-1 e PoP-2), como foi observado em [Drago et al. 2012]. Observamos também que colaborações externas são as atividades dominante nas redes residenciais. Esses resultados fornecem indícios de que serviços populares de armazenamento na nuvem, como Dropbox, são utilizados intensivamente para realização de trabalhos colaborativos. Tal comportamento é diferente em serviços menos populares, como o Ubuntu One, onde apenas 1,8% dos usuários realizam trabalhos colaborativos [Gracia-Tinedo et al. 2015].

Ainda com relação aos estados de atividade dos usuários (Figura 4) observa-se que a fração de usuários colaborando em redes nos campi é pelos menos duas vezes superior a cada um dos demais estados. Nas residências, a fração de colaborações em rede é aproximadamente metade da fração de colaborações externas, aproximando-se da fração de usuários com foco em armazenamento. Contudo, as Figuras 4(c-d) mostram uma tendência leve de crescimento de colaborações nas residências, ao passo que a atividade com foco em armazenamento diminui. Esses resultados mostram que os provedores de serviço devem se preparar para atender a uma maior demanda por colaborações em redes via arquiteturas alternativas de sincronização, que diminuem custos com largura de banda [Gonçalves et al. 2015].

A Figura 5 mostra o volume das sincronizações, i.e., volumes de *upload* e *download*, dos usuários em cada estado ao longo do tempo nas redes monitoradas. O vo-

lume de sincronizações em todas as redes apresenta flutuações devido a características de cauda pesada dos fluxos de dados dos usuários [Gonçalves et al. 2014]. Essas flutuações são mais intensas em residências devido à maior frequência de sincronização de arquivos multimídia volumosos (e.g., fotos e vídeos) [Gonçalves et al. 2016]. Apesar dessas flutuações, pode-se observar novamente que colaborações em redes é a atividade com tráfego dominante nos campi. Nas residências, diferentemente da fração de usuários mostrada na Figura 4(c-d), a fração de sincronizações para colaborações em redes se aproxima da fração de sincronizações para colaborações externas. Observa-se também que a fração de sincronizações dos usuários com foco em armazenamento é a menor na maioria das semanas em todas as redes.

Em resumo, nessa seção observamos que trabalhos colaborativos tem um impacto significativo no Dropbox em termos de número de usuários e volume de sincronizações em redes como universidades e ISPs. Pelo menos 55%(22%) dos usuários nos campi(residências) realizam ao menos uma colaboração em rede semanalmente, ao passo que, para colaborações externas essa porcentagem é pelo menos de 20%(47%). Em relação ao tráfego gerado por sincronizações, os usuários que colaboram em rede representam pelo menos 41%(9%) do tráfego do serviço nos campi(residências). Considerando os usuários com colaborações externas essa porcentagem é pelo menos 10%(35%).

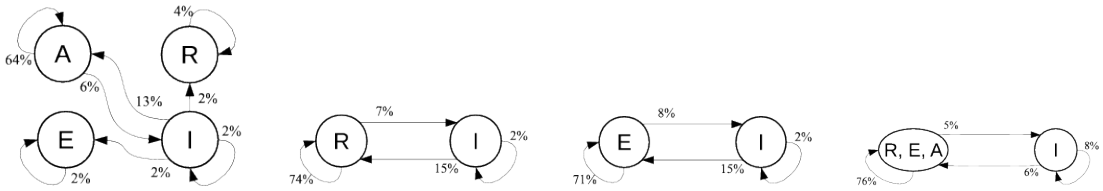
5.2. Perfis de Usuários

Um mesmo usuário pode estar em diferentes estados de atividade ao longo do tempo. Nesta seção, nós analisamos a dinâmica da transição dos usuários entre os estados de atividades definidos na seção anterior. Nesse sentido, nós identificamos perfis de usuários considerando a maneira que eles realizam essas transições e caracterizamos esses perfis.

Para identificar perfis de usuários, nós calculamos, para cada usuário, a frequência de transições entre os estados de atividade: colaborando em rede (R), colaborando com usuários externos (E), armazenamento (A) e inatividade (I) que significa o não armazenamento de dados na nuvem. A seguir, consideramos cada usuário como um vetor de dezesseis elementos (i.e., possibilidades de transições entre os quatro estados), e executamos o algoritmo K-means para agrupar usuários similares quanto às transições de atividade. Esse algoritmo requer a escolha do número de grupos k (i.e., *clusters*). Nós empregamos diferentes medidas de qualidade de agrupamento para escolher o valor de k, como coeficientes de variação intra, inter *clusters* e beta-CV, além de inspeção visual dos centróides dos agrupamentos. Baseado nessas medidas chegamos à conclusão que $k = 3$ grupos é o suficiente para representar os perfis de atividade dos usuários.

A Figura 6 representa cada um dos perfis identificados (i.e., centróides do algoritmo K-means) através de grafos direcionados. Em cada grafo os vértices são os estados de atividade e as arestas são as porcentagens (ou probabilidades) de um usuário realizar uma transição entre dois estados de atividade diferentes ou permanecer no mesmo estado. Para melhor clareza na figura, omitimos transições com porcentagens menores que 2%. Nós identificamos perfis de usuários para as quatro redes analisadas, mas mostraremos resultados apenas para Campus-2 e PoP-2 representando respectivamente os campi e residências, dado que obtivemos conclusões semelhantes para Campus-1 e PoP-1.

As Figuras 6(a-c) mostram os três perfis de atividade dos usuários no Campus-2. Note que cada perfil tem um estado de atividade predominante. Por exemplo, a Figura 6(a) mostra um perfil de usuários com foco em armazenamento, i.e., 64% das transições



(a) Campus-2 Perfil A (b) Campus-2 Perfil R (c) Campus-2 Perfil E (d) PoP-2 Perfis R, E e A

Figura 6. Perfis de atividade dos usuários nomeados em cada item pelo estado predominante em cada perfil: foco em armazenamendo dados (A), compartilhamentos na rede (R), compartilhamentos externos (E) e inativo (I).

Tabela 2. Caracterização dos perfis de usuários: mediana(média) do número de parcerias, compartilhamentos e volume sincronizado para usuários do perfil

	Campus-2		PoP-2	
Perfil	Compartilhamentos	Vol. Sincronizado (MB)	Compartilhamentos	Vol. Sincronizado (MB)
A	1 (1,1)	3,9 (267,9)	1 (1,1)	41,6 (450,9)
R	3 (3,8)	20,4 (253,3)	2 (3,4)	70,7 (492,8)
E	1 (1,9)	11,8 (212,0)	2 (2,4)	82,2 (846,0)

* Perfil determinado pela atividade predominante.

dos usuários consistem em permanecer na atividade A. Os perfis (b) e (c) mostram que usuários realizando trabalhos colaborativos tem uma probabilidade menor de transitar para o estado de inatividade (7-8%), e dificilmente permanecem nesse estado (apenas 2% das transições). A porcentagem de usuários que o algoritmo K-Means atribuiu para cada perfil se aproxima das proporções que medimos na seção anterior (Figura 4 (a-b)). Os perfis das Figuras 6 a, b e c (i.e., A, R e E) tem respectivamente 7%, 67% e 26% dos usuários.

Na Figura 6(d) mostramos uma única imagem para exemplificar os três perfis de atividade do usuário no PoP-2. Nas redes residenciais, usuários nos estados R, E ou A tendem a permanecer no mesmo estado (76% das transições) ou transitam para o estado I (5% das transições). Note que, diferentemente dos campi, os usuários nas residências tendem a permanecer inativos por mais tempo (8% das transições). A porcentagem de usuários nos perfis onde predominam as atividades A, R ou E são respectivamente 21%, 28% e 51%. Nota-se uma porcentagem menor para o perfil de usuários colaborando dentro das redes, corroborando com os resultados apresentados nas Figuras 4(c-d).

Finalmente, caracterizamos os três perfis de atividade do usuário identificados acima. Novamente, Campus-2 e PoP-2 representam respectivamente as redes dos campi e residências. A Tabela 2 mostra a mediana (e média) do número de parcerias, *namespaces* e volume sincronizado por usuários dentro de cada perfil. Para os campi, pode-se observar que o perfil R representa os usuários que impõem maior carga em volume de sincronizações no sistema. Em valores medianos, os usuários nesse perfil sincronizam 20MB e 3 *namespaces* por mês. Contudo, valores médios de sincronizações por mês superiores a 200MB indicam que há usuários com volume de dados muito acima dos valores típicos em todos os perfis.

Para os perfis residenciais, a Tabela 2 mostra que o volume de sincronizações é superior aos respectivos perfis nos campi. Possivelmente, isso ocorre devido à maior sincronização de conteúdo multimídia como fotos e vídeos nas residências. Outra

observação interessante é que o volume mediano de sincronizações no perfil R se aproxima em cerca de 86% do perfil E (majoritário em redes residenciais). Isso mostra que a colaboração entre usuários em uma mesma residência ou sob o mesmo ISP é expressiva, mesmo que esse perfil ainda não seja majoritário nessas redes.

Em resumo, identificamos três perfis de atividade (semanal) do usuário considerando trabalhos colaborativos em sistemas de armazenamento na nuvem em redes de universidades e ISPs residenciais: colaborações dentro da rede monitorada (R), colaborações externas às redes (E) e foco em armazenamento de dados (A). Os perfis R e E são majoritários em redes universitárias e residenciais respectivamente, mostrando que serviços populares como Dropbox vem sendo usados intensivamente para trabalhos colaborativos ao invés de puramente armazenamento de dados na nuvem.

6. Conclusões

Neste artigo apresentamos uma investigação sobre a relação entre trabalho colaborativo e o nível de atividade dos usuários no Dropbox, um dos serviços de armazenamento na nuvem mais populares atualmente. Nossa principal conclusão é que tal serviço vem sendo utilizado pelos usuários não apenas para armazenamento e *backup* de dados, mas principalmente para realização de trabalhos colaborativos via compartilhamento de dados.

Primeiramente mostramos que a atividade de um usuário em serviços de armazenamento na nuvem pode ser medida pela sua quantidade de colaborações. Observamos correlações altas e não lineares entre o nível de atividade do usuário (i.e., volume de dados sincronizados com a nuvem) e o seu número de compartilhamentos (*namespaces*). Observamos também que a média de sincronizações por usuário em compartilhamentos tende a aumentar em relação ao número de usuários nesses compartilhamentos. Esses resultados podem ser explorados pelos provedores de serviços, que ao aprimorarem as funcionalidades de colaboração/compartilhamento de dados entre usuários em seus serviços, podem aumentar as suas receitas decorrente do aumento da atividade dos usuários.

A seguir, mostramos que trabalhos colaborativos tem um impacto significativo no uso do Dropbox em termos de número de usuários e volume de sincronizações em grandes redes como universidades e ISPs residenciais. Observamos que, semanalmente, pelo menos 55% dos usuários nos campi realizam ao menos uma colaboração em rede, ao passo que, pelo menos 47% dos usuários em residências realizam ao menos uma colaboração com usuários externos ao seu provedor de Internet. Com base nessas análises identificamos três perfis de usuários que consideram colaborações internas ou externas às redes monitoradas e usuários que focam basicamente em armazenamento de dados na nuvem.

Trabalhos futuros incluem investigações de causalidade na relação entre crescimento de colaborações em rede e aumento de atividade do usuário e vice-versa. Pretendemos também construir modelos para quantificar e predizer a importância ou o peso de trabalhos colaborativos em diferentes redes, assim como funções de utilidade para usuários que indicam o quanto serviços de armazenamento na nuvem é importante para os mesmos e quando vale a pena pagar por esse serviço.

Referências

Amrehn, P., Vandenbroucke, K., Hossfeld, T., Moor, K. D., Hirth, M., Schatz, R., and Casas, P. (2013). Need for Speed? On Quality of Experience for Cloud-based File Storage Services. In *Proc. of PQS*.

Bocchi, E., Drago, I., and Mellia, M. (2015a). Personal Cloud Storage Benchmarks and Comparison. *IEEE Transactions on Cloud Computing*, PP(99):1–14.

Bocchi, E., Drago, I., and Mellia, M. (2015b). Personal Cloud Storage: Usage, Performance and Impact of Terminals. In *Proc. of IEEE CloudNet*.

Casas, P. and Schatz, R. (2014). Quality of Experience in Cloud Services: Survey and Measurements. *Comput. Netw.*, 68(1):149–165.

Chen, F., Mesnier, M. P., and Hahn, S. (2014). Client-Aware Cloud Storage. In *Proc. of MSST*.

Costa, E., Costa, L., Drago, I., Vieira, A., Ziviani, A., da Silva, A. P. C., and de Almeida, J. M. (2014). Análise da Topologia Social do Dropbox. In *Proc. of WP2P+*.

Cowell, F. (2011). *Measuring Inequality*. Oxford University Press.

Cui, Y., Lai, Z., Wang, X., Dai, N., and Miao, C. (2015). QuickSync: Improving Synchronization Efficiency for Mobile Cloud Storage Services. In *Proc. of MobiCom*.

Drago, I., Mellia, M., Munafò, M. M., Sperotto, A., Sadre, R., and Pras, A. (2012). Inside Dropbox: Understanding Personal Cloud Storage Services. In *Proc. of IMC*.

Gonçalves, G., Drago, I., da Silva, A. P. C., de Almeida, J. M., and Vieira, A. (2014). Caracterização e Modelagem da Carga de Trabalho do Dropbox. In *Proc. of SBRC*.

Gonçalves, G., Drago, I., Vieira, A., Silva, A., Almeida, J., and Mellia, M. (2016). Workload models and performance evaluation of cloud storage services. *Computer Networks*. `http://dx.doi.org/10.1016/j.comnet.2016.03.024`.

Gonçalves, G., Drago, I., Vieira, A. B., da Silva, A. P. C., and de Almeida, J. M. (2015). Analyzing the Impact of Dropbox Content Sharing on an Academic Network. In *Proc. of SBRC*.

Gracia-Tinedo, R., Artigas, M. S., Moreno-Martinez, A., Cotes, C., and Lopez, P. G. (2013). Actively Measuring Personal Cloud Storage. In *Proc. of CLOUD*.

Gracia-Tinedo, R., Tian, Y., Sampé, J., Harkous, H., Lenton, J., García-López, P., Sánchez-Artigas, M., and Vukolic, M. (2015). Dissecting ubuntuone: Autopsy of a global-scale personal cloud back-end. In *Proc. of IMC*.

Jain, R. K. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, New York, USA.

Li, Z., Jin, C., Xu, T., Wilson, C., Liu, Y., Cheng, L., Liu, Y., Dai, Y., and Zhang, Z.-L. (2014). Towards Network-Level Efficiency for Cloud Storage Services. In *Proc. of IMC*.

Shami, N. S., Muller, M., and Millen, D. (2011). Browse and discover: Social file sharing in the enterprise. In *Proc. of CSCW*.

Wang, H., Shea, R., Wang, F., and Liu, J. (2012). On the Impact of Virtualization on Dropbox-Like Cloud File Storage/Synchronization Services. In *Proc. of IWQoS*.

Zhang, Y., Dragga, C., Arpaci-Dusseau, A. C., and Arpaci-Dusseau, R. H. (2014). ViewBox: Integrating Local File Systems with Cloud Storage Services. In *Proc. of FAST*.

Caracterização do Serviço de Táxi a partir de Corridas Solicitadas por um Aplicativo de *Smartphone*

Átila M. Silva Júnior[1], Miguel L. M. Sousa[1], Faber Z. Xavier[1], Wender Z. Xavier[1], Jussara M. Almeida[2], Artur Ziviani[3], Francisco Rangel[4], Cláudio Ávila[4], Humberto T. Marques-Neto[1]

[1] Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
30.535-901 - Belo Horizonte - Brasil

[2]Universidade Federal de Minas Gerais (UFMG)
31.270-010 - Belo Horizonte - Brasil

[3]Laboratório Nacional de Computação Científica (LNCC/MCTI)
25.651-075 - Petrópolis - Brasil

[4]WayTaxi - Belo Horizonte - Brasil

{atila.junior,miguel.sousa,wender.xavier}@sga.pucminas.br

jussara@dcc.ufmg.br,ziviani@lncc.br, {faber,humberto}@pucminas.br

{francisco.rangel,claudio.avila}@waytaxi.com.br

Abstract. *The use of mobile apps to support a plethora of services offers a valuable opportunity to study population dynamics in urban areas. Somehow, this can guide improvements of the mobile Internet access as well as of these systems. In particular, data obtained from mobile applications for requesting taxi rides allows for the study of human mobility. Moreover, the analysis of the user behavior patterns of such services enables valuable insights of the user needs. In this paper, we present a characterization study of the taxi rides from mobile apps data. In total, we analyze 37,183 rides requested by 16,442 users and serviced by 3,663 distinct taxi drivers in Belo Horizonte during one week. We find, for instance, that 51% of the rides have been requested in the southern part of downtown and among the cancellations, 49% occurred within the first minute of waiting before the taxi attendance.*

Resumo. *O uso de aplicativos de smartphones gera uma grande quantidade de dados sobre a movimentação de pessoas em grandes centros urbanos. Isto cria possibilidades de avaliação da mobilidade humana que, de certa forma, pode direcionar evoluções da infraestrutura da rede de acesso à Internet móvel bem como desses sistemas. Os aplicativos utilizados para realização de chamadas de táxi são exemplos de fontes de dados capazes de subsidiar estudos sobre a mobilidade humana e sobre o comportamento de usuários desses sistemas. Neste trabalho foi realizada uma caracterização de corridas de táxi solicitadas através de um aplicativo de smartphone na cidade de Belo Horizonte. Foram analisadas 37.183 corridas solicitadas por 16.442 passageiros e atendidas por 3.663 taxistas durante uma semana. Pôde-se constatar que 51% das corridas foram solicitadas em uma área específica da cidade e dentre os cancelamentos, 49% foram realizados até o primeiro minuto de espera.*

1. Introdução

A complexidade das condições de tráfego em grandes centros urbanos exige que abordagens eficientes sejam adotadas para monitorar e apoiar os processos de tomada de decisão, sobretudo em relação à mobilidade urbana. O crescente fluxo de veículos nessas grandes cidades tem complicado, de certa forma, a mobilidade urbana, causando atrasos, estresse e insatisfação na população. Algumas aplicações para dispositivos móveis (e.g., Moovit, Waze e Uber)[1] que utilizam a geo-localização de seus usuários oferecem possibilidades para melhoria dessa mobilidade e, por sua vez, geram um grande volume de dados que, quando analisados corretamente, permitem uma melhor compreensão do comportamento social e do tráfego de uma cidade [Zheng et al. 2014].

O táxi é um dos serviços de transporte público mais utilizados em todo o mundo e tem como característica importante a flexibilidade para escolha do seu trajeto. Aplicações para *smartphones* (e.g., EasyTaxi ou WayTaxi)[2] estão sendo amplamente adotadas pelos usuários de táxi para solicitar corridas com mais eficiência. Geralmente, para utilizar esses serviços, tanto o passageiro quanto o taxista acionam a rede de dados móvel e o GPS do seu dispositivo. O passageiro inicia o processo informando sua localização em seu *smartphone* para solicitar uma corrida. Em seguida, a prestadora de serviço responsável pelo aplicativo encaminha a solicitação para os taxistas disponíveis que estão a uma determinada distância do passageiro. Ao primeiro taxista que aceitar a requisição será atribuída a corrida. Algumas características como reputação dos usuários, dos motoristas, e tipo de veículo também podem ser consideradas por esse tipo de aplicação.

O produto da análise de dados desse tipo de aplicação pode direcionar evoluções para as companhias de táxi, para as operadoras de telefonia celular e também para o poder público. As companhias de táxi conhecendo melhor a demanda pelo serviço e o comportamento dos usuários poderão melhorar a qualidade do serviço prestado, bem como oferecer novos serviços. As operadoras de telefonia celular, por outro lado, poderão incrementar o planejamento de capacidade de sua infraestrutura de rede em regiões onde se registra um grande uso do serviço. O poder público também poderá usufruir desse tipo de análise para melhorar o planejamento urbano e assim dimensionar a quantidade de licenças necessárias para atender a demanda pelo serviço de táxi em determinada localidade.

Neste trabalho a mobilidade humana foi analisada a partir dos dados associados às corridas de táxis solicitadas por meio de uma aplicação de *smartphone* que oferece o serviço de localização de táxis em diversas cidades brasileiras. Assim, este estudo se difere de muitos outros realizados recentemente [Castro et al. 2013, Qi et al. 2013, Zhang et al. 2011], principalmente, por não analisar a dinâmica das corridas de táxi exclusivamente a partir de dados gerados por GPS utilizados por vários motoristas. Ou seja, aqui, a mobilidade humana foi avaliada usando dados de corridas requisitadas e aceitas por uma aplicação de *smartphone* que, além de usar o GPS deste tipo de dispositivo, possui outras informações, como por exemplo a localização inicial do passageiro e tempo entre o seu pedido e o início da corrida. Desta forma, analisou-se não apenas os locais e horários de embarque e desembarque, mas as informações espaço-temporais de chamada do passageiro, de aceite do taxista, de desembarque e de eventuais cancelamentos realizados pelos usuários.

[1] `moovitapp.com`, `waze.com`, `uber.com`

[2] `easytaxi.com`, `waytaxi.com`

Especificamente, os dados analisados foram gerados pela aplicação móvel de solicitação de corridas de táxi denominada *WayTaxi*. As corridas avaliadas foram realizadas na cidade de Belo Horizonte-MG, que possui, aproximadamente, uma população de 2,5 milhões habitantes em sua região metropolitana. Apesar da presença de outros serviços similares com grande poder de mídia, o *WayTaxi* é atualmente um dos mais populares aplicativos de solicitação de táxis nesta cidade. No total, analisou-se 37.183 corridas solicitadas por 16.442 passageiros atendidos por 3.663 taxistas distintos durante a primeira semana de março de 2015. O estudo de caracterização abrange aspectos temporais e espaciais relacionados com as solicitações de aceite e cancelamento de corridas, bem como o tempo de espera dos passageiros. Com base nos resultados, observou-se que sexta-feira e sábado foram os dias com a maior demanda pelo serviço e que os principais destinos dos passageiros foram pontos de saída da cidade, como a estação rodoviária e o ponto de ônibus que realiza conexão direta com os aeroportos da cidade.

O restante deste estudo está organizado como descrito a seguir. Os trabalhos relacionados estão presentes na Seção 2. Em seguida, a base de dados analisada e a metodologia de caracterização foram apresentados nas Seções 3 e 4, respectivamente. Os principais resultados foram discutidos na Seção 5. Conclusões e trabalhos futuros foram apresentados na Seção 6.

2. Trabalhos Relacionados

Os dados da movimentação de táxis em uma determinada região são uma fonte de informação valiosa para a caracterização da mobilidade urbana [Zheng et al. 2014]. Estes dados podem ser utilizados para classificar regiões típicas de embarque e desembarque de passageiros, para descobrir e avaliar pontos de interesse [Wan et al. 2013] da cidade, bem como as interconexões entre esses locais [Zhang et al. 2012]. A coleta de informações da posição e do tempo a partir dos dispositivos presentes nos automóveis permitem tanto o monitoramento do deslocamento populacional [Ganti et al. 2013] quanto a avaliação das condições do tráfego [Moosavi and Hovestadt 2013].

Para analisar o comportamento dos taxistas e classificar suas estratégias como eficientes ou não, três aspectos são considerados: a procura por passageiros, a entrega dos mesmos nos seus destinos e a área de atuação do taxista [Zhang et al. 2015]. Entretanto, dentre eles, o primeiro, cujo objetivo é a busca por novos clientes, foi o mais explorado em outros estudos. Em [Li et al. 2011] foi proposta uma técnica para avaliar a eficiência na descoberta dos passageiros. Já em [Yuan et al. 2011] foi proposto um sistema de recomendação para taxistas com base no padrão de mobilidade dos passageiros e nos embarques realizados. No trabalho de [Tang et al. 2013] foi usado o Processo de Decisão de Markov para recomendar aos taxistas onde encontrar passageiros mais lucrativos e em [Qu et al. 2014] foi desenvolvida uma técnica que determina as rotas mais rentáveis.

Logo, tendo posse dos dados de monitoramento do serviço de táxi, é possível realizar estudos que avaliem o comportamento da população, o fluxo de veículos nas vias urbanas e o próprio serviço de transporte [Castro et al. 2013]. Ou seja, com base nos parâmetros de geolocalização, duração e tarifação das corridas, é possível compreender melhor o funcionamento desse tipo de sistema e propor melhorias que o torne mais eficiente [Veloso et al. 2011].

Diferente dos trabalhos previamente realizados, que em sua maioria focam na

análise de dados coletados dos dispositivos de GPS, o presente estudo mostrou a avaliação de diferentes aspectos de padrões da mobilidade humana. Com os dados gerados pela aplicação avaliada, diferente dos trabalhos previamente realizados, foi possível realizar uma análise espaço-temporal das chamadas dos usuários e de seus momentos de cancelamento. Isso permite a realização de análises relacionadas ao tempo de espera do passageiro assim como das regiões onde a demanda não foi atendida pelos taxistas.

3. Base de Dados

Um aplicativo típico utilizado para realização de uma chamada de táxi permite ao passageiro localizar um taxista próximo que esteja disponível e, além disto, acompanhar seu percurso até o local combinado para embarque. Como pré-requisitos para o funcionamento desse sistema, tanto o passageiro quanto o taxista devem possuir acesso à rede de dados e permitir o uso do GPS do seu dispositivo por parte da versão da aplicação especifica para o seu respectivo tipo do usuário (passageiro ou taxista). Após a chamada, esse tipo de aplicativo permite que tanto o passageiro quanto o taxista cancelem, respectivamente o pedido ou o aceite da corrida, geralmente, antes do seu início.

A base de dados analisada consiste em informações anonimizadas referentes a corridas de táxi realizadas na cidade de Belo Horizonte-MG. Os dados foram disponibilizados pela empresa *WayTaxi*, detentora de um dos aplicativos utilizados para chamada de táxi mais populares nessa cidade. As principais funcionalidades dessa aplicação de *smartphone* são a requisição, o aceite e o cancelamento de uma corrida de táxi, as quais também estão presentes em outros aplicativos, tais como, o EasyTaxi e o 99Taxi [3]). A base de dados utilizada neste estudo possui os seguintes dados:

1. Identificador único da corrida;
2. Identificador único e anônimo do passageiro;
3. Identificador único e anônimo do motorista;
4. Data e Hora de solicitação da corrida;
5. Data e Hora de aceite da corrida;
6. Data e Hora de conclusão da corrida;
7. Data e Hora de cancelamento da corrida;
8. Latitude e Longitude do passageiro;
9. Latitude e Longitude do taxista;
10. Latitude e Longitude da conclusão da corrida.

Esta base de dados contém todas as corridas realizadas por intermédio do *WayTaxi* na primeira semana de março de 2015 em Belo Horizonte-MG[4]. O intervalo de tempo analisado corresponde a uma típica semana de trabalho desta cidade, pois corresponde ao semestre letivo e não houve nenhum feriado. Durante este período foram realizadas 37.183 corridas, solicitadas por 16.442 passageiros distintos. Esses passageiros foram atendidos por 3.663 motoristas distintos, valor que representa 54% dos 6.756 taxistas credenciadas pela Empresa de Transportes e Trânsito de Belo Horizonte (BHTrans) no período avaliado. Em seguida, será apresentada a metodologia para caracterização desta base de dados.

[3] `http://www.99taxis.com/`

[4] O acesso à esses dados será avaliado sob demanda encaminhada para humberto@pucminas.br.

4. Metodologia de Caracterização

A metodologia proposta tem como objetivo caracterizar a dinâmica do uso de uma aplicação para requisição do serviço de táxi com o propósito de fornecer informações sobre a mobilidade humana na área urbana considerada. A caracterização abrange aspectos temporais, vide Seção 4.1, e aspectos espaciais, vide Seção 4.2, do serviço de transporte avaliado.

4.1. Caracterização Temporal

As seguintes questões norteiam a primeira etapa da metodologia de caracterização proposta: (i) *Quais foram os períodos com maior demanda (em termos de solicitações) por corridas de táxi?*, (ii) *Tal demanda foi atendida pelos taxistas?*, (iii) *Quanto tempo os passageiros estão dispostos a aguardar para serem atendidos? Ou seja, depois de quanto tempo os passageiros cancelam uma solicitação de táxi?*, (vi) *Qual a frequência de utilização da aplicação por parte dos seus usuários?* e (v) *Quanto tempo após o cancelamento o usuário realiza uma nova chamada?*. Essas questões foram respondidas com o objetivo de fornecer um panorama geral da utilização da aplicação e do comportamento de seus usuários no decorrer de uma semana.

Primeiramente, foram observados os momentos de maior demanda pelo serviço e se ela foi atendida pelos taxistas cadastrados. Sendo assim, os momentos de requisição dos passageiros, de aceite do taxista e conclusão das corridas foram analisados. A partir desses dados, avaliou-se quais foram os dias da semana e horas do dia com maior percentual de requisições, atendimentos e conclusões das corridas.

Em seguida, as corridas solicitadas que foram canceladas foram analisadas. Os momentos de maior concentração dos cancelamentos podem revelar períodos específicos de possível insatisfação dos usuários. Engarrafamentos e quantidade insuficiente de táxis disponíveis são alguns motivos que podem justificar a ocorrência dos cancelamentos em determinados períodos do dia.

Os momentos de cancelamento também foram utilizados para realizar uma análise de tolerância dos usuários. Para tanto, realizou-se uma comparação entre o tempo de espera do passageiro até o atendimento e o tempo decorrido entre as chamadas e os cancelamentos. A partir dessa comparação, pode-se avaliar quanto tempo o passageiro estaria disposto a esperar além do habitual para que o seu pedido seja aceito por um taxista.

Depois, analisou-se o intervalo de tempo entre a conclusão e a solicitação de uma nova corrida realizada pelo usuário da aplicação. Dessa forma, variou-se a janela de tempo de um minuto até uma semana e verificou-se o percentual de corridas que foram solicitadas pelo mesmo passageiro e o percentual de corridas que foram atendidas pelo mesmo taxista para cada janela de tempo. Com isso, pôde-se observar a frequência que os usuários utilizaram o serviço em cada período de tempo avaliado.

De maneira similar, avaliou-se o período de tempo entre os cancelamentos e a realização de uma nova solicitação. Dessa forma, pôde-se observar o comportamento do passageiro após os cancelamentos realizados antes e depois do aceite do taxista, se ele/a realizou uma nova chamada imediatamente ou se esperou alguns instantes.

4.2. Caracterização Espacial

Na segunda etapa da metodologia de caracterização, buscou-se responder as seguintes questões: (i) *Quais foram as regiões da cidade com maior demanda pelo serviço de táxi?*, (ii) *Onde estavam os taxistas quando aceitaram as solicitações dos passageiros?*, (iii) *Onde estavam os passageiros quando cancelaram as corridas?*, (iv) *Quais foram os principais destinos dos passageiros?* e (v) *Quais as distâncias entre a posição de chamada e as posições de aceite, cancelamento e conclusão?*. A caracterização espacial utiliza como referência a localização geográfica associada às ações dos passageiros e dos taxistas.

Quando um passageiro solicita um táxi pelo *smartphone*, ele/a fornece sua localização de embarque e a aplicação envia a solicitação para os taxistas localizados a uma determinada distância do passageiro. Ao analisar as coordenadas geográficas associadas a um grande número de corridas, é possível identificar as áreas da cidade que apresentaram maior demanda por esse serviço. Dessa forma, essa informação poderia ser usada, por exemplo, para uma redistribuição eficiente da frota de táxi conforme a demanda, a fim de reduzir os tempos de resposta e deslocamento, bem como evitar cancelamentos de serviços.

Quando a solicitação do passageiro é atendida, a aplicação mostra a localização do taxista. O passageiro então é capaz de acompanhar, em tempo real, a rota tomada pelo táxi em direção ao ponto de embarque. De posse de tais informações, juntamente com a localização dos pontos de táxi da cidade, foi possível determinar se o taxista estava na região de um ponto de táxi ou se ele estava circulando pela cidade quando a solicitação foi aceita. A Empresa de Transporte e Trânsito de Belo Horizonte (BHTrans) fornece os endereços completos de todos os pontos de táxi da cidade. Usando a API *Google Maps Geocoding* [5], realizou-se a geo-codificação desses endereços e se obteve as coordenadas geográficas de cada ponto de táxi. Com base em tais posições, foi determinado quais corridas foram atendidas por taxistas localizados a uma determinada distância dos pontos de táxi oficiais da cidade.

A corrida finaliza com o desembarque do passageiro ou com o cancelamento após o atendimento. Analisando a concentração das corridas concluídas que não foram canceladas, identificou-se os principais destinos dos passageiros de táxi da cidade. Esses lugares mostram também onde grande quantidade dos taxistas ficariam disponíveis para atender novas solicitações de serviço.

Como se pode ver na Seção 3, o posicionamento do passageiro ao realizar o cancelamento não foi disponibilizado. Desta forma, considerou-se a posição de chamada apenas das corridas que foram canceladas antes do aceite do taxista. Sendo assim, pôde-se observar quais foram os lugares de maior concentração de usuários insatisfeitos (ou impacientes), ou seja, onde os táxis não conseguem chegar em tempo hábil para atender os passageiros.

Finalmente, foram avaliadas as distâncias entre a posição de chamada e as posições de atendimento, cancelamento e conclusão de uma mesma corrida. Para tanto, avaliou-se o percentual de cada evento (atendimento, cancelamento e conclusão) que foi realizado a uma determinada distância da posição de chamada. Na Seção 3 se pode ver

[5] `https://developers.google.com/maps/documentation/geocoding/intro`

que não existe o campo específico para posição de cancelamento, mas, como as corridas canceladas após o aceite do taxista possuem a latitude e longitude do táxi no campo referente ao posicionamento de conclusão da corrida se pode fazer tal avaliação.

5. Resultados

Nesta Seção os resultados obtidos com a aplicação da metodologia proposta serão discutidos. Em primeiro lugar, na Seção 5.1, avalia-se a demanda pelo serviço de táxi com base nos aspectos temporais da base de dados. Em seguida, na Seção 5.2, os aspectos espaciais são analisados com intuito de construir uma visão abrangente do sistema de táxi da cidade Belo Horizonte.

5.1. Análise Temporal

Como mencionado na Seção 3, a base dados é composta por: data e hora de chamada, atendimento, conclusão e cancelamento. Com o objetivo de obter um panorama inicial desses dados, as corridas foram separadas por dia e em seguida foi realizada a contagem da ocorrência de cada evento. Como se pode ver na Figura 1, de domingo até quinta-feira o número de chamadas foi em torno de 4.000 para cada dia. Por outro lado, na sexta e no sábado esses valores praticamente dobraram evidenciando uma demanda maior pelo serviço de táxi nesses dias. Mais precisamente, as chamadas de táxi de sexta-feira e de sábado correspondem a 43,7% do total de chamadas analisadas.

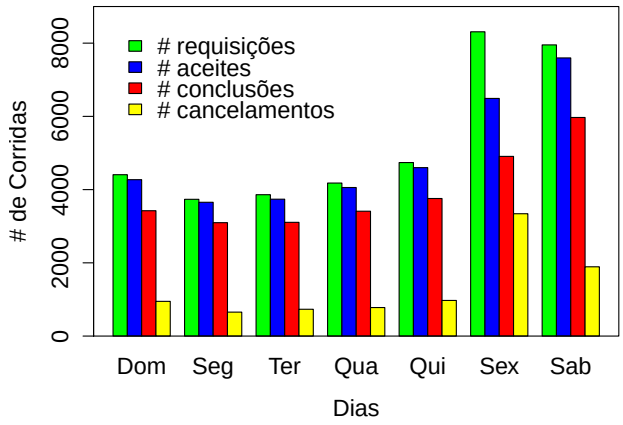


Figura 1. Distribuição de corridas por dia.

No entanto, pode-se observar que nem todas as corridas solicitadas foram atendidas, sobretudo no dia em que foi registrada a maior demanda. Isso se deve à ocorrência de cancelamentos realizados pelos usuários, que podem desistir das corridas antes de serem atendidos ou após algum tempo após o aceite do taxista. Sendo assim, ao todo, 25,2% das corridas foram canceladas, sendo que 7,4% foram antes do atendimento e 17,7% após o momento que o taxista aceitou a corrida.

Do total de chamadas realizadas na sexta-feira, 40,9% não chegaram a ser concluídas. Desse valor, 21,9% foram canceladas antes mesmo do aceite do taxista. Ao realizar a comparação de sexta-feira com sábado pode-se perceber que apesar de ambos os dias possuírem quantidades semelhantes de chamadas, no sábado o percentual de cancelamentos foi menor. Isso pode estar relacionado ao trânsito, uma vez que a sexta-feira, último

dia de trabalho da semana, normalmente apresenta mais congestionamento de trânsito fazendo com que as corridas demorem mais e os taxistas fiquem impossibilitados de atenderem novos chamados. Após identificar os dias de maior demanda, foi analisada cada hora do dia. Dessa forma, pôde-se ter uma visão mais detalhada do comportamento desse serviço em cada momento do dia.

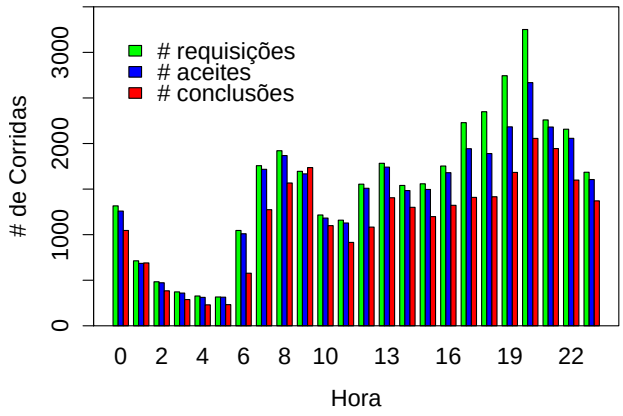


Figura 2. Distribuição de corridas por hora.

Como se pode ver na Figura 2 o período de maior demanda foi registrado às 20 horas. Todavia, nota-se que entre as 13 e as 21 horas grande parte das solicitações também não foram atendidas pelos taxistas. Outros pequenos picos podem ser observados às 6 da manhã, início do expediente, e às 11 horas, início do horário de almoço. Nesses momentos os taxistas atenderam rápido, mas, demoraram um pouco mais para chegar aos passageiros ou, quando chegaram, os passageiros não se encontravam mais no lugar combinado para embarque. Isso pode ser constatado pelo fato do número de corridas concluídas ser menor que as aceitas pelos taxistas.

A Figura 1 mostra que sexta-feira apresentou uma grande quantidade de cancelamentos. Logo, foi realizado um estudo mais aprofundado nesse dia avaliando os cancelamentos realizados antes e depois do aceite do taxista. Quando o cancelamento ocorre antes do atendimento de um taxista, certamente terá sido efetuado pelo passageiro. Por outro lado, se tal cancelamento for realizado após o aceite do taxista, ambos usuários podem ter disparado essa ação. Um taxista pode cancelar uma corrida por diversos motivos: ele pode ter encontrado um passageiro a caminho do local de embarque, ele pode ter ficado retido no trânsito e não ter conseguido chegar no local demarcado em tempo hábil, ou pode ter chegado ao local e não ter encontrado o passageiro.

Quando se observa a Figura 3 percebe-se que durante boa parte das horas do dia os cancelamentos ocorreram após o aceite do taxista. Porém, às 17, 18 e 19 hs, a maioria dos cancelamentos foram registrados antes do atendimento, ou seja, pelo passageiro. Logo, se tratando de um horário de grande demanda, pode-se constatar que nesse momento muito provavelmente os táxis estavam ocupados ou impossibilitados de atender novas chamadas.

Em seguida, foi analisado o tempo de espera do passageiro. Para tanto, avaliou-se as corridas que foram aceitas pelo taxista e as canceladas pelos passageiros antes do aceite. São fatores que podem levar ao cancelamento do passageiro: pode não haver táxis próximos disponíveis, ele pode ter optado por outra forma de transporte (e.g. Uber ou

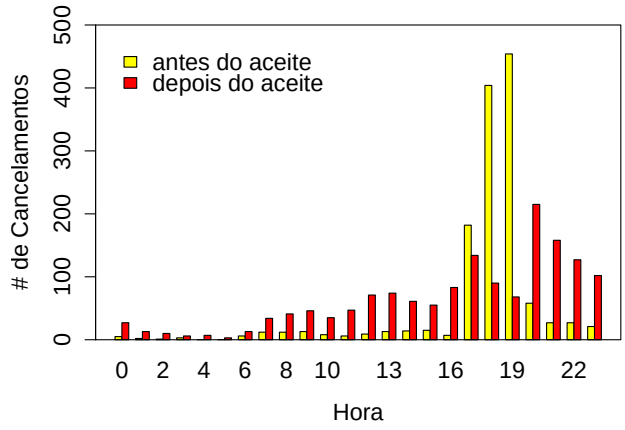


Figura 3. Corridas canceladas na sexta-feira.

transporte coletivo), ele pode ter solicitado um táxi utilizando outro aplicativo do gênero que o atendeu antes do aplicativo em análise ou ele pode, até mesmo, ter conseguido outro táxi vazio perto de onde está.

A Figura 4 mostra a distribuição acumulativa do tempo de espera do passageiro considerando as corridas aceitas pelos taxistas, bem como aquelas que foram canceladas antes do aceite. Pode-se observar que aproximadamente metade de todos os cancelamentos (49,5%) foram realizados até o primeiro minuto após a solicitação. Um outro rápido crescimento no número de cancelamentos ocorreu entre o quinto e oitavo minuto de espera. Ou seja, o passageiro pode ter se cansado de esperar ou optado por outro meio de transporte.

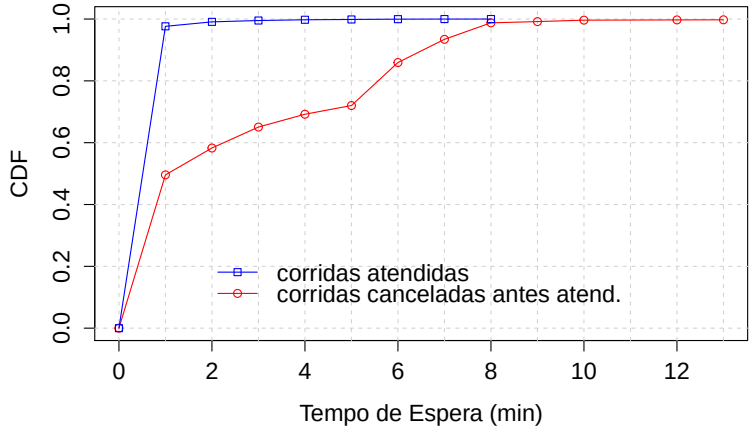


Figura 4. Tempo de espera do passageiro.

Por outro lado, a Figura 4 também mostra que quase todas as corridas atendidas (97,6%) foram aceitas por algum taxista até o primeiro minuto de espera. Esse tempo é curto porque corresponde ao momento em que o taxista aceita a solicitação do passageiro em seu *smartphone*. Em média, o intervalo de tempo entre a solicitação e o cancelamento da corrida foi de 3,6 minutos. Em contraste, o tempo médio de espera entre a solicitação do passageiro e o aceite do taxista foi de apenas 28 segundos. Portanto, o passageiro, baseado em experiências passadas, pode ter se tornado mais exigente a respeito do tempo de resposta do serviço.

Também foi avaliado o intervalo de tempo entre a conclusão de uma corrida e uma nova solicitação do serviço tanto para o passageiro quanto para o taxista. Observa-se na Figura 5 que a curva referente aos taxistas tem um grande crescimento entre uma (26%) e doze horas (56%). Isto mostra que em grande parte das corridas os taxistas esperaram até doze horas para atenderem uma nova solicitação realizada através do aplicativo. Durante este intervalo o taxista pode ter atendido outras corridas utilizando outro aplicativo ou da maneira convencional, sem o uso de aplicativos. Pode-se observar também que 87% dos taxistas atenderam uma nova solicitação até o período de uma semana.

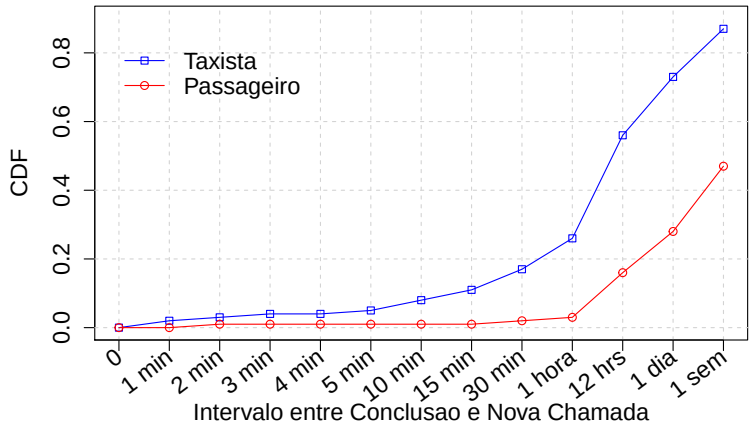


Figura 5. Intervalo de tempo entre a conclusão de uma corrida e a solicitação de uma nova corrida para o passageiro e atendimento de uma chamada para o taxista.

Ainda na Figura 5, ao observar a curva referente ao intervalo de tempo entre chamadas dos passageiros, nota-se um crescimento entre uma e duas horas de intervalo, saltando de 3% para 16%. Este intervalo de tempo pode corresponder aos usuários que utilizaram o serviço de chamada de táxi para ir e voltar do local desejado ou para dar continuidade a uma corrida interrompida para uma atividade no meio do caminho. Nota-se ainda que no período de uma semana, 47% dos passageiros realizaram uma nova chamada após uma conclusão. Isso revela que grande parte dos passageiros realizou apenas uma chamada durante o período avaliado.

Por fim, de maneira similar, foi avaliado o intervalo entre o cancelamento e uma nova chamada realizada pelo passageiro. Ao observar a Figura 6, nota-se que, tanto para os cancelamentos realizados antes quanto para os efetuados após o aceite, grande parte dos passageiros esperaram até um minuto para realizar uma nova chamada. Isso, mais uma vez, revela a impaciência do passageiro ao utilizar esse tipo de serviço, fazendo com que ele cancele e efetue novas chamadas até ser atendido.

5.2. Análise Espacial

Após os momentos de maior demanda serem evidenciados, foi realizada uma análise espacial com base no posicionamento geográfico dos usuários do serviço de táxi. Como as latitudes e longitudes de chamada, aceite e conclusão foram disponibilizadas, a demanda de cada evento será avaliada a fim de evidenciar os lugares de maior concentração dos usuários desse aplicativo. Essas posições foram obtidas com base na rede de dados móveis e do GPS dos *smartphones* do taxista e do passageiro. A posição de chamada é

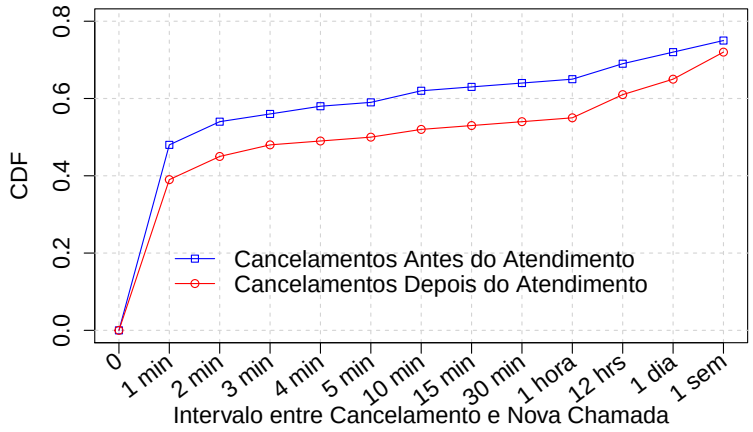
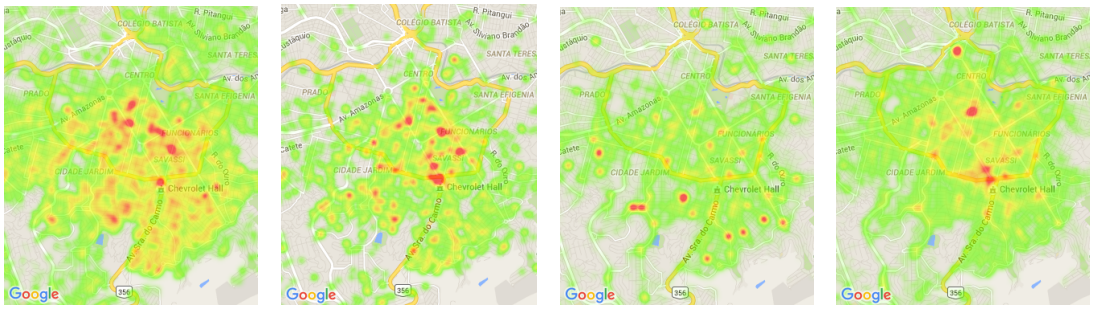


Figura 6. Intervalo de tempo entre os cancelamentos antes e após o aceite e a solicitação de uma nova corrida.

obtida quando o passageiro solicita uma corrida. A posição de atendimento é registrada quando o taxista aceita atender o passageiro e a posição de conclusão é obtida quando o passageiro chega ao seu destino.

Durante o período de avaliação foram registradas corridas de todas as regiões da cidade. No entanto, grande parte dessas corridas de táxi se concentraram em torno da região central. Isso se deve ao fato de essa ser a área de maior concentração populacional, onde se situam edifícios públicos e comerciais. Logo, para fins de ilustração, foi construído um mapa de calor com foco na região central para cada posição da corrida. O uso desse tipo de mapa é interessante para mostrar as regiões de concentração de determinados eventos. Sendo assim, como se pode ver, o mapa da Figura 7(a) mostra a concentração das chamadas, o da Figura 7(b) mostra a concentração das chamadas canceladas antes do atendimento, o mapa da Figura 7(c) a concentração dos atendimentos e o da Figura 7(d) a concentração das conclusões.



(a) Chamadas (b) Cancelamentos antes do aceite (c) Atendimentos (d) Conclusões

Figura 7. Mapa de calor das posições das chamadas aceitas e canceladas, atendidas e concluídas.

A posição de chamada revela as regiões de maior demanda pelo serviço de táxi. Quando se observa a Figura 7(a), pode-se notar que as chamadas estão concentradas sobretudo na região centro-sul da cidade. Após a realização da geo-codificação reversa das coordenadas constatou-se que, de fato, 51,14% das corridas foram solicitadas nessa área.

Essa região, além de possuir uma grande concentração populacional, é caracterizada por conter os bairros de maior padrão de vida da cidade.

Como as posições do cancelamento não foram disponibilizadas, o posicionamento dos cancelamentos realizados antes do aceite foi estimado com base na posição de chamada das corridas. Essa posição mostra onde o passageiro estava esperando quando ele desistiu da corrida. Ao observar a Figura 7(b), nota-se que os cancelamentos se concentraram nos lugares onde foram registradas as maiores demandas, região centro-sul.

Por outro lado, a posição de aceite mostra como os taxistas estão distribuídos pela cidade. Na Figura 7(c), diferente das posições de chamada que se manifestaram como borrões disformes, pode-se ver concentrações em formato de pontos. Isso é justificável pelo fato de que os táxis, quando vazios, se concentram nos pontos de táxis espalhados pela cidade. A relação dos endereços de cada um desses pontos é disponibilizada pela empresa de transporte e de trânsito da cidade. Sendo assim, utilizou-se dessas posições para obter as corridas que foram atendidas pelos taxistas que estavam nesses pontos. Para tanto, foi delimitada uma região de 60 metros em torno de cada posição e calculou-se o percentual de atendimentos efetuados nessas regiões. Constatou-se assim que 29,37% das corridas foram atendidas nas proximidades dos pontos de táxi. O restante dos atendimentos ocorreu quando o taxista estava circulando por diferentes pontos da cidade.

As posições de maior concentração das conclusões revelam quais são os principais destinos das pessoas que utilizaram o táxi no período avaliado. Ao observar a Figura 7(d) nota-se três regiões de maior concentração, das quais duas são caracterizadas por serem lugares de saída da cidade e a outra como sendo um centro comercial onde se concentram bares, restaurantes e muitas lojas. O ponto de calor localizado mais ao norte do mapa é a rodoviária. Já o ponto central é referente a uma linha de ônibus que tem como destino os aeroportos da cidade. Na região de calor localizada mais ao sul se localiza o bairro Savassi, caracterizado como região nobre da cidade.

Por fim, foram avaliadas as distâncias entre a posição de chamada e as posições dos eventos de atendimento, cancelamento e conclusão das corridas. Os resultados foram apresentados na Figura 8. Ao observar a curva referente às distâncias entre as chamadas e os atendimentos percebe-se um crescimento praticamente linear até 1 km de distância. Observa-se também que 95% dos atendimentos estão a essa distância das chamadas. Isso mostra que não adianta o passageiro aumentar o raio de busca para mais de um quilômetro que dificilmente algum taxista o atenderá. Nota-se também que os taxistas que atenderam as chamadas estavam próximos dos passageiros, uma vez que 77% estavam a uma distância menor ou igual a 500 metros do ponto de embarque quando aceitaram atender o passageiro.

Ao observar a curva referente à distância entre as chamadas e os cancelamentos, nota-se uma quantidade considerável de cancelamentos que ocorreram quando o taxista estava próximo ao local demarcado pelo passageiro para embarque. Do total de corridas canceladas após o aceite, em 37% o taxista estava a uma distância de até 100 metros do ponto de embarque e 52% a uma distância de 200 metros. Tais cancelamentos podem ter sido realizados pelo taxista quando ele chegou ao local demarcado e não encontrou o passageiro.

Finalmente, a curva que mostra as distâncias entre as chamadas e as suas respec-

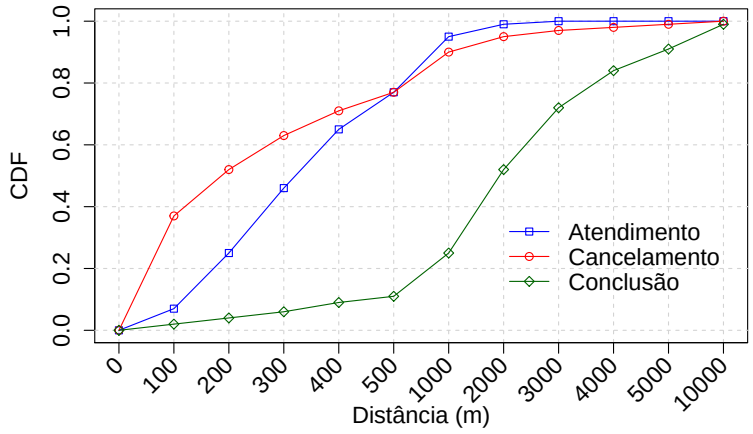


Figura 8. Distância entre a posição de chamada e os eventos de atendimento, cancelamento e conclusão.

tivas conclusões retrata a distância total das corridas. Sendo assim, observa-se que 52% das corridas concluídas tiveram uma distância de até 2 km. Ou seja, grande parte dos passageiros usa o táxi para percorrerem distâncias não muito longas nessa cidade.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi realizada uma caracterização das corridas de táxi solicitadas por uma aplicação para *smartphones*. Esses dados correspondem a 37.183 corridas solicitadas por 16.442 passageiros e atendidas por 3.663 taxistas da cidade de Belo Horizonte-MG por um período de uma semana. A partir da caracterização descobriu-se que a demanda não foi atendida nos momentos de pico, resultando em cancelamentos dos usuários. Constatou-se também que os passageiros, provavelmente ansiosos para serem atendidos, cancelaram as corridas no primeiro minuto e realizaram novas solicitações em seguida. Em geral, as corridas foram solicitadas na região centro-sul da cidade, atendidas por taxistas que estavam próximos dos passageiros e finalizadas em um bairro nobre e lugares que possibilitam a saída da cidade. Isso indica regiões da cidade e horários que possuem mais demanda de acesso à Internet móvel de alta velocidade.

Possíveis trabalhos futuros incluem a aplicação da metodologia em bases de dados mais significativas, coletadas em um período de tempo maior, bem como a proposta de métodos para uma distribuição mais eficiente dos táxis pela cidade a fim de reduzir o tempo de espera dos passageiros e, consequentemente, o percentual de cancelamentos. Outra possível direção seria a realização do paralelo entre o padrão de mobilidade dos táxis com outros modais, tais como, ônibus e bicicletas.

7. Agradecimentos

Este trabalho foi financiado pelo PROBIC da PUC Minas, FAPEMIG, FAPERJ, CAPES, CNPq, BigSea, FAPEMIG-PRONEX-MASWeb – Modelos, Algoritmos e Sistemas para a Web (processo APQ-01400-14) e INCTWeb (MCT/CNPq 573871/2008-6).

Referências

Castro, P. S., Zhang, D., Chen, C., Li, S., and Pan, G. (2013). From taxi GPS traces to social and community dynamics: A survey. *ACM Computing Surveys*, 46(2):17:1–17:34.

Ganti, R., Srivatsa, M., Ranganathan, A., and Han, J. (2013). Inferring human mobility patterns from taxicab location traces. In *Proc. of the ACM Int. Conf. on Ubiquitous Computing (UbiComp)*, pages 459–468.

Li, B., Zhang, D., Sun, L., Chen, C., Li, S., Qi, G., and Yang, Q. (2011). Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *Proc. of the IEEE Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 63–68.

Moosavi, V. and Hovestadt, L. (2013). Modeling urban traffic dynamics in coexistence with urban data streams. In *Proc. of the ACM Int. Workshop on Urban Computing (UrbComp)*, pages 10:1–10:7.

Qi, G., Pan, G., Li, S., Wu, Z., Zhang, D., Sun, L., and Yang, L. (2013). How long a passenger waits for a vacant taxi – large-scale taxi trace mining for smart cities. In *Proc. of IEEE Int. Conf. on Green Computing and Communications (GreenCom) and Internet of Things (iThings/CPSCom)*, pages 1029–1036.

Qu, M., Zhu, H., Liu, J., Liu, G., and Xiong, H. (2014). A cost-effective recommender system for taxi drivers. In *Proc. of the ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 45–54.

Tang, H., Kerber, M., Huang, Q., and Guibas, L. (2013). Locating lucrative passengers for taxicab drivers. In *Proc. of the ACM Int. Conf. on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 504–507.

Veloso, M., Phithakkitnukoon, S., and Bento, C. (2011). Urban mobility study using taxi traces. In *Proc. of the Int. Workshop on Trajectory Data Mining and Analysis (TDMA)*, pages 23–30.

Wan, X., Gao, M., Kang, J., and Zhao, J. (2013). Taxi origin-destination areas of interest discovering based on functional region division. In *Proc. of the Int. Conf. on Innovative Computing Technology (INTECH)*, pages 365–370.

Yuan, J., Zheng, Y., Zhang, L., Xie, X., and Sun, G. (2011). Where to find my next passenger. In *Proc. of the ACM Int. Conf. on Ubiquitous Computing (UbiComp)*, pages 109–118.

Zhang, D., Li, N., Zhou, Z.-H., Chen, C., Sun, L., and Li, S. (2011). iBAT: Detecting anomalous taxi trajectories from gps traces. In *Proc. of the ACM Int. Conf. on Ubiquitous Computing (UbiComp)*, pages 99–108.

Zhang, D., Sun, L., Li, B., Chen, C., Pan, G., Li, S., and Wu, Z. (2015). Understanding taxi service strategies from taxi GPS traces. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):123–135.

Zhang, W., Li, S., and Pan, G. (2012). Mining the semantics of origin-destination flows using taxi traces. In *Proc. of the ACM Int. Conf. on Ubiquitous Computing (UbiComp)*, pages 943–949.

Zheng, Y., Capra, L., Wolfson, O., and Yang, H. (2014). Urban computing: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38:1–38:55.

Classificando Comportamentos Sociais em Redes Veiculares

Davidysson A. Alvarenga[1], **Felipe D. Cunha**[2], **Aline C. Viana**[3]
Raquel A. F. Mini[1], **Antônio A. F. Loureiro**[2]

[1] DCC – Pontifícia Universidade Católica de Minas Gerais (PUC-Minas)
[2]DCC – Universidade Federal de Minas Gerais (UFMG)
[3]INRIA – France

davidysson@ufmg.br, fdcunha@dcc.ufmg.br, aline.viana@inria.fr
raquelmini@pucminas.br, loureiro@dcc.ufmg.br

Abstract. *Vehicular networks are networks comprised by vehicles traveling cities and highways. During their trajectories, these vehicles interact with other vehicles in order to make safer and enjoyable traffic. These interactions may be influenced by several factors. To mention a few: vehicle speed, roads condition and time of day. Moreover, driver behavior and its interests can influence in many vehicle features. In this context, the Vehicular Social Networks arise as a new perspective to vehicular networks. These are networks in which vehicles "socialize" and share common interests. In this work, we evaluate the behavior of vehicles in two mobility scenarios, in order to classify them according to the interactions performed, identifying common interests and similar routines. Thus, we use metrics of complex networks and statistical techniques. Results prove the existence of routines and human features in Vehicular Networks.*

Resumo. *Redes veiculares são redes formadas por veículos que transitam nas cidades e rodovias. Durante suas trajetórias estes veículos interagem uns com outros veículos com o objetivo de tornar o trafego mais seguro e a viagem agradável. Essas interações podem ser influenciadas por diversos fatores. Dentre eles pode-se citar: a velocidade do veículo, as condições das vias e o período do dia. Além disso, o comportamento do condutor e seus interesses podem influenciar em muitas características do veículo. Nesse contexto, surgem as Redes Sociais Veiculares como uma nova perspectiva para as redes veiculares. Essas são redes nas quais os veículos "socializam" e compartilham interesses comuns. Neste trabalho serão avaliados o comportamento dos veículos em dois cenários de mobilidade, com o objetivo de classificá-los de acordo com as interações realizadas, identificando interesses comuns e rotinas semelhantes. Para isso, foram utilizadas métricas de redes complexas e técnicas estatísticas. Resultados comprovam a existência de rotinas e propriedades sociais nas Redes Veiculares.*

1. Introdução

Redes Veiculares (*Vehicular Networks* - VANETs) são um tipo de rede formadas por veículos, que interagem entre si diretamente ou por meio de infraestrutura instaladas nas vias públicas. Esses veículos possuem capacidade de processamento, comunicação e sensoriamento. No entanto, devido à grande mobilidade dos veículos, estas redes possuem uma alta variação na sua topologia. Outros fatores que cooperam para esta variação são os limites de velocidade e direção impostos aos veículos pelas vias públicas.

Durante suas trajetórias os veículos interagem entre si, e diversos fatores podem influenciar nestas interações. Dentre eles podemos citar: o horário do dia, a velocidade dos veículos, o local, engarrafamentos, a existência de eventos, etc. Por exemplo, em horários de pico (tráfego denso), as vias públicas podem ser vistas como um lugar popular, onde é mais suscetível ocorrer interações. Outro fator que pode influenciar nas interações é o comportamento do condutor e seus interesses comuns com outros condutores. Este traz consigo uma nova perspectiva para as redes veiculares: as Redes Sociais Veiculares (*Vehicular Social Networks* - VSNs) [25].

Neste contexto, este trabalho se propõe a investigar as VSNs, estudando a influência do comportamento humano na mobilidade veicular, identificando padrões, rotinas semelhantes e interesses comuns. Além disso, pretende-se discutir as melhorias que esse conhecimento pode oferecer no estabelecimento das redes veiculares. Para isso, foram analisadas duas bases de dados de veículos usando métricas de redes complexas e técnicas estatísticas. Os resultados apontam que é possível identificar classes de comportamentos nas redes sociais veiculares.

O restante do trabalho está organizado da seguinte forma: a Seção 2 apresenta um breve resumo dos trabalhos relacionados encontrados na literatura. A Seção 3 descreve a metodologia utilizada neste trabalho, as bases de dados selecionadas e as métricas e técnicas utilizadas. A Seção 4 apresenta os resultados quantitativos, de acordo com cada métrica avaliada. A Seção 5 discute sobre como aplicar estas métricas na elaboração de novos serviços para VANETs. Finalmente, a Seção 6 apresenta as conclusões e as perspectivas futuras.

2. Trabalhos Relacionados

Estudos recentes tem comprovado a influência do comportamento humano na mobilidade dos veículos [9, 12, 5]. Nesses estudos pode-se averiguar como as interações ocorrem entre os veículos, a frequência dessas interações, os horários de maior probabilidade de ocorrência e também como utilizar melhor essas interações em projetos de novos serviços e aplicações para redes veiculares.

Normalmente esses estudos se apoiam na teoria de redes complexas, na análise de traces de mobilidade de veículos e em técnicas estatísticas [21]. Assim, os autores exploram a avaliação das métricas em grafos que representam contatos entre os veículos numa determinada janela de tempo de avaliação. A Tabela 1 apresenta os trabalhos que caracterizaram cenários de mobilidade veicular e que buscam entender melhor a evolução topológica da rede.

Pode-se observar que em [9, 20, 15], os autores avaliaram apenas pequenos cenários analíticos, nos quais foram utilizados geradores sintéticos de mobilidade. De forma mais ampla, o comportamento urbano das grandes cidades foi estudado em [18, 23, 16], nos quais os autores avaliaram as características da mobilidade realística dos veículos que circulam dentro de alguns municípios.

Por outro lado, em [13, 12] os autores estudaram cenários com a mobilidade de táxis reais em algumas cidades, extraindo propriedades e comportamentos da rede. Entretanto, por questões de privacidade, na literatura não se encontra dados reais de veículos particulares, o que pode impossibilita uma melhor extração de comportamentos humanos. Desta forma, em [22], os autores constroem e analisam a estrutura da rede utilizando fotografias de algumas regiões das cidades.

Autores	Traces Avaliados	Métricas Avaliadas	Observações
Fiore e Härri (2008) [9]	Cenários analíticos	Duração de encontros, densidade, grau dos vértices, *clusters* e coeficiente de agrupamento	modelos distintos; raio fixo na comunicação; não utiliza grafos temporais e aleatórios; área geográfica delimitada; duração limitada.
Pallis et (2009) [18]	Zurique	Grau dos vértices, duração de encontros, diâmetro, centralidade de vértices, *clusters*, coeficiente de agrupamento, comunidades e correlação entre as métricas	Varia o raio da comunicação, não utiliza grafos temporais e aleatórios; Avalia apenas algumas horas do dia; área geográfica delimitada.
Loulloudes, Pallis e Dikaiakos (2010) [13]	Xangai e Los Angeles	grau dos vértices, diâmetro, densidade, centralidade de vértices, duração e frequência de encontros, *clusters*, coeficiente de agrupamento e comunidades	Mobilidade realística em Los Angeles; não varia o raio da comunicação; não utiliza grafos temporais e aleatórios; área geográfica delimitada; duração limitada.
Rezende et al. (2011) [20]	Cenários analíticos	Diâmetro, coeficiente de agrupamento e grau dos vértices	Região urbana/rodovia; densidade fixa; não varia o raio de comunicação; não compara com grafos aleatórios; varia o tamanho da janela temporal; área geográfica delimitada; duração limitada.
Liu et al. (2012) [12]	São Francisco e Xangai	Grau dos vértices, distância média e coeficiente de agrupamento	Seleciona apenas o maior componente do grafo; não varia o raio da comunicação; compara com grafos aleatórios; não teve avaliação de janelas temporais.
Uppoor, Naboulsi e Fiore (2012-2013) [23, 16]	Colônia	Densidade e distribuição de veículos, tempo de viagem, distância percorrida, duração de encontros componentes, grau dos vértices, centralidade de intermediação e assortatividade	Analiza apenas sobre o tráfego de veículos; varia o raio de comunicação; compara com grafos aleatórios; não varia o tamanho da janela temporal, duração limitada.
Thakur, Hui e Helmy (2013) [22]	Connecticut, Londres, Seattle, Sydney, Toronto e Washington	Coeficiente de agrupamento, densidade, grau dos vértices e distância média	Modelagem dos grafos utilizando *webcams* como vértices e as vias como arestas; não avalia variação no raio de comunicação; compara com grafos aleatórios; não varia o tamanho da janela temporal; avalia apenas quatro horários do dia.
Monteiro et al. (2013) [15]	Cenários analíticos	Grau dos vértices, distância média, coeficiente de agrupamento e conectividade	Região urbana/rodovias; Variação de densidade; Raio de comunicação fixo; Não utiliza grafos temporais e aleatórios; área geográfica delimitada; duração limitada.
Cunha et al. (2013-2014) [5, 4]	São Francisco e Zurique	Grau dos vértices, distância média, coeficiente de agrupamento, densidade e persistência de arestas	Raio de comunicação fixo; compara com grafos aleatórios; não varia tamanho da janela temporal.

Tabela 1. Trabalhos recentes encontrados na literatura que apresentam caracterização da mobilidade veicular.

Contudo, parte destes trabalhos apresentam algumas limitações. Alguns não analisaram as bases em sua plenitude, uma vez que isolaram a área central das cidades, ignorando as regiões periféricas. Outros computaram poucas métricas, não relacionando os resultados com outros modelos. Estudos recentes mostraram que, com a análise de todas essas bases de dados, é possível encontrar características humanas no comportamento dos veículos [5, 4]. Apesar destas análises estarem apoiadas em bases de dados de táxis ou mesmo em bases sintéticas, foi possível verificar a existência de interesses comuns e comportamentos similares em uma rede veicular. Esses comportamentos reforçam a ideia das Redes Sociais Veiculares [25], redes nas quais os veículos interagem, compartilham interesses comuns e comportamentos semelhantes.

3. Metodologia

Nesta seção serão apresentados os detalhes de toda a metodologia utilizada na avaliação das bases de dados. O foco dessa avaliação é entender melhor como acontecem as interações entre os veículos, destacando os comportamentos e as características sociais dos mesmos. Em primeiro momento, será descrito os detalhes das bases de dados avaliadas. Em seguida, será descrito o processo de geração dos grafos e as métricas utilizadas.

3.1. Bases de Dados Avaliadas

A seleção das bases de dados utilizadas neste trabalho foi definida por critérios que envolveram: mobilidade diversificada ao longo das 24 horas do dia, registros em mais de um dia da semana, granularidade semelhante, quantidade semelhante de veículos em circulação por faixas horárias e disponibilidade pública. A primeira base de dados foi obtida a partir da extração de coordenadas de localização geográfica durante a simulação de um modelo de mobilidade realística na cidade de Helsinque. A segunda base de dados contém dados reais da mobilidade de táxis dentro da cidade de Roma.

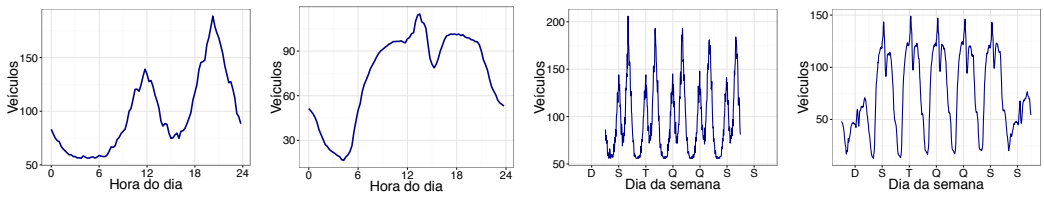
Parâmetro	Valores
Pessoas	2 mil
Grupos	8 grupos em 4 distritos
Carro próprio	50%
Locais	Casa, trabalho e áreas de lazer
Meios de transporte	Ônibus, carro ou táxis
Simulador	The ONE
Filtro utilizado	Apenas veículos

Tabela 2. Parâmetros do modelo *Working Day Movement Model*.

- **Helsinque:** representa a mobilidade realística gerada pelo modelo *Working Day Movement Model* (WDM) [7]. Assim, a mobilidade de 2.000 pessoas foi gerada utilizando veículos diversos (carros, ônibus ou táxis), se locomovendo entre 3 locais distintos (residência, trabalho e pontos de lazer), ao longo das 24 horas do dia, dentro de um intervalo de 1 semana útil. A simulação do tráfego foi realizada no *software* The ONE, utilizando os parâmetros descritos na Tabela 2.
- **Roma:** esta base de dados contém a mobilidade real de 320 táxis que trabalham na região central da cidade. Os registros foram coletados durante 4 semanas em 2014 [2].

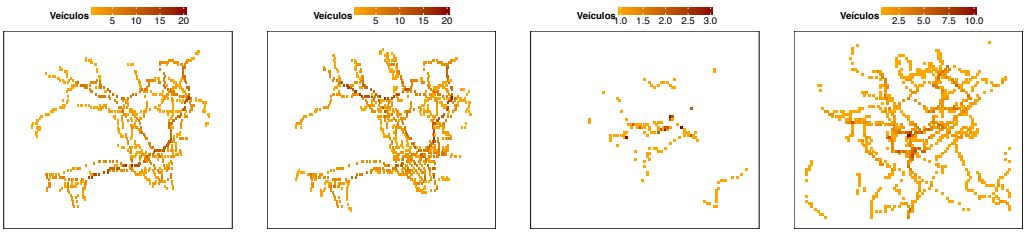
A Figura 1 apresenta a evolução do tráfego diário e semanal para as duas bases. É possível observar um trânsito maior durante os horários de pico, com uma redução no horário de almoço e nas madrugadas. Além disso, para a base de Helsinque é possível verificar que não existe tráfego durante os finais de semana. Este é um modelo que considera apenas dias úteis, desprezando finais de semana e feriados. Ao olhar para o tráfego semanal em Roma, nota-se uma redução aos finais de semana, o que pode ser explicado pelo contexto dos dias.

Analisando a evolução do tráfego sob as cidades, a Figura 2 apresenta o *heatmap* do tráfego. Pode-se destacar Roma, onde às 6 am. o tráfego é bem esparso, e o mesmo apresenta-se muito mais denso às 6 pm, onde os veículos passam a utilizar vias paralelas aos congestionamentos. Além disso, nota-se que as duas cidades possuem características de tráfego bem distintas para os mesmos horários do dia. Esse é um comportamento que indica características peculiares em cada cidade, e que as mesmas devem ser consideradas no projeto de soluções para essas redes.



(a) Helsinque diário. (b) Rome diário. (c) Helsinque semanal. (d) Roma semanal.

Figura 1. Evolução do tráfego ao longo do dia e da semana.



(a) Helsinque 6am. (b) Helsinque 6pm. (c) Roma 6am. (d) Roma 6pm.

Figura 2. Mapa de calor representando a mobilidade de veículos em horários de pico.

3.2. Grafos Temporais

Um grafo temporal no tempo t é um grafo não dirigido, definido como um grafo $G(t) = (V, E)$, onde V representa o conjunto com todos os veículos V_i e E representa o conjunto de arestas E_{ij}. Em $G(t)$, existe uma aresta $E_{ij}(t)$ entre os veículos V_i e V_j durante o tempo t, se $i \neq j$. A avaliação de cada métrica deve levar em conta cada um dos grafos temporais $G(t)$, com as interações que acontecem em cada período t. Por exemplo, se t estiver definido para 15 minutos, os registros ocorridos entre $08{:}00$ – $08{:}14$ serão agrupados em t_0 e os registros ocorridos entre $08{:}15$ – $08{:}29$ horas serão agrupados em t_1.

Durante a criação dos grafos temporais, o critério para inserção de arestas entre dois vértices é definido pela distância entre as coordenadas dos veículos. Se essa distância for menor ou igual ao valor do raio da comunicação, os vértices serão conectados. Todas as arestas que pertencerem ao mesmo intervalo de avaliação serão agrupadas em um grafo $G(t)$. Na direção de entender melhor as características destas arestas e verificar as propriedades existentes, cada grafo $G(t)$ será comparado a outro grafo aleatório G^R gerado pelo algoritmo de urna [11], com mesma quantidade de arestas e distribuição de graus dos vértices. Por meio deste processo, foi possível verificar a existência de estruturas sociais no grafo temporal, que é inexistente no grafo aleatório.

3.3. Métricas Avaliadas

Nesta seção serão descritas as métricas escolhidas para avaliação das bases de dados. Dentre as métricas conhecidas de redes complexas [3], neste trabalho optou-se por agrupá-las de forma que as métricas que representam o mesmo comportamento pertençam a mesma categoria. Assim, quatro grupos foram definidos: métricas que mensuram a influência dos vértices sob seus vizinhos, métricas relacionadas ao fenômeno "Mundo Pequeno", métricas de centralidade e métricas de semelhança de contatos. Os valores finais de cada métrica foram calculados utilizando o valor médio observado em cada intervalo de tempo analisado.

Vértices vs. Vizinhos: O grau do vértice representa a quantidade de interações que o mesmo faz com sua vizinhança. Além disso, vértices que apresentam grau elevado são os mais ativos na rede, e podem atuar como um canal para a troca de informações [6]. Tal métrica pode ser vista como uma medida de centralidade, indicando a popularidade do vértice no grafo. Em um nível global, um alto grau pode indicar uma alta concentração de veículos próximos uns aos outros, comum em horários de pico ou engarrafamentos. Por outro lado, o grau médio dos vizinhos (*average neighbor degree*) representa a média dos graus de todos vizinhos que cada vértice possui. A avaliação da sua correlação com os

graus dos vértices mostra a tendência de um vértice se ligar a outros com graus similares. Nesse sentido, a rede é considerada como assortativa quando a correlação for crescente. Nessas redes, veículos que compartilham rotinas estarão ligados uns aos outros e com graus semelhantes [17].

Outra importante métrica que se pode avaliar no contexto de comportamentos semelhantes é a presença de comunidades no grafo. No contexto de redes veiculares, uma comunidade representa indivíduos que possuem interesses comuns. No grafo pode-se destacar um subconjunto de vértices fortemente conectados entre si. Neste trabalho, para avaliação dessa métrica optou-se em executar o algoritmo *Walktrap* [19], aplicado em cada componente existente no grafo temporal $G(t)$ a fim de detectar conjunto de veículos que formam as comunidades. Ao quantificar as conexões entre os veículos, pode-se avaliar também a densidade de arestas no grafo. Essa pode apresentar valores elevados em regiões urbanas em relação às áreas periféricas ou rurais, devido ao tráfego lento ou à existência de engarrafamentos. Valores baixos de densidade ajudam a identificar regiões cuja conectividade pode ser interrompida mais facilmente, exigindo estratégias que busquem minimizar tal problema.

Centralidade: A centralidade de um vértice no grafo indica o quão importante aquele vértice é em relação aos demais. Do ponto de vista de distância e comunicação, o quão mais central um vértice for, mais caminhos mínimos passarão por ele e mais comunicações o mesmo irá intermediar. Uma forma de definir essa centralidade é a métrica centralidade de intermediação (*beetweeness centrality*), que representa a frequência com que um veículo faz parte de um caminho mais curto entre quaisquer dois outros veículos. Essa métrica é importante em situações nas quais dois vértices não são vizinhos e dependem dos outros vértices para realizar sua troca de dados. A posição da intermediação indicará a influência do vértice ao levar em consideração o controle exercido por ele sobre as interações entre os dois outros.

Mundo Pequeno: O fenômeno "mundo pequeno" *(small-world)* [14] define a premissa que qualquer pessoa pode se conectar à outra por meio de uma sequência de relacionamentos em um grau médio de 5 a 6 pessoas [14, 26], o que numa rede representaria o número de conexões que separam duas pessoas. Além disso, redes que apresentam esse fenômeno possuem uma grande tendência de formar comunidades, ou seja, formar agrupamentos de nós que compartilham interesses comuns. Para avaliar e classificar uma rede como "mundo pequeno" utiliza-se as métricas coeficiente de agrupamento e a distância média. Assim, essas redes tendem a apresentar uma distância média baixa e um elevado coeficiente de agrupamento. A distância média representa o número de saltos necessários para a conexão entre um par quaisquer de veículos. Valores altos podem indicar atrasos em transmissões de dados, pois os veículos estão fisicamente distantes, fora do raio da comunicação, e provavelmente não visitam locais próximos. Por outro lado, valores baixos indicam lugares próximos ou pontos em comum em suas rotas, podendo representar interesses comuns. Considerando o coeficiente de agrupamento, nas redes veiculares, os veículos que possuem interesses comuns tendem a ficar mais agrupados, o que contribui para um aumento no valor do coeficiente de agrupamento.

Semelhança de Contatos: No contexto das redes sociais veiculares, a semelhança de contatos pode ser avaliada por duas métricas: a sobreposição topológica e a persistência de arestas. A sobreposição topológica (*topological overlap*) retrata a tendência de existir vértices com vizinhos partilhados, enquanto que a persistência de arestas (*edge persistence*) define o número de vezes que dois vértices se encontram na mesma janela temporal,

ao longo do período. Esta métrica permite identificar as relações regulares entre entidades. De acordo com [24], a avaliação destas métricas permite a classificação das interações em quatro tipos de relacionamentos. Essa classificação baseia-se na comparação com grafos aleatórios, de forma que é possível quantificar se os relacionamentos em uma rede veicular são aleatórios ou obedecem a um padrão. Como observado na Tabela 3, na classe de Amigos (*friends*), são encontradas características sociais sobre rotinas semelhantes entre os veículos tanto na persistência de arestas quanto na sobreposição topológica. Na classe de Conhecidos (*acquaintance*), essa característica é encontrada somente na sobreposição topológica. Na classe de Pontes (*bridges*), a característica social é encontrada somente na persistência de arestas. E finalmente, na classe Aleatória (*random*), não há presença de tais características nas duas métricas.

Classes	Sobreposição Topológica	Persistência de Arestas
Amigos	Social	Social
Conhecidos	Social	Aleatório
Pontes	Aleatório	Social
Aleatória	Aleatório	Aleatório

Tabela 3. Classificação da semelhança entre os contatos [24].

4. Resultados

Nesta seção serão apresentados os resultados da análise efetuada sobre as bases de dados. Para essa análise foi considerada a metodologia e as métricas definidas na Seção 3. Primeiramente, a Seção 4.1 define os parâmetros e detalhes da análise. Em seguida, os resultados serão apresentados de acordo com as propriedades definidas na Seção 3.3.

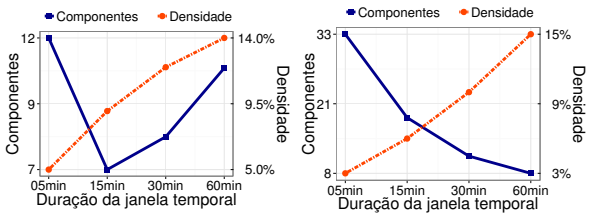
4.1. Parâmetros

A análise dos dados foi efetuada considerando os valores ao longo do dia e da semana. O objetivo principal desta análise é identificar comportamentos e propriedades que descrevem rotinas e interesses dos condutores numa rede veicular. A partir desse entendimento será possível propor serviços e protocolos que atendam melhor a demanda dos seus usuários.

Com o intuito de verificar o impacto da avaliação nas interações entre os veículos, neste trabalho o raio de comunicação foi configurado de acordo com definições do protocolo 802.11p. No entanto, foi avaliada uma variação no raio da comunicação, com valores de 100 e 150 metros, a fim de verificar o impacto nos resultados, uma vez que um incremento em seu valor pode agregar mais contatos antes inalcançáveis.

Considerando a janela de tempo para agregação de contatos, foram considerados os seguintes valores: 5, 15, 30 e 60 minutos, conforme discussões realizadas em trabalhos relacionados [8, 10]. Ao considerar valores pequenos de janela, foi possível encontrar um grafo mais esparso com muitos componentes. Em contrapartida, janelas de tempo com grande intervalo agrupam uma grande quantidade de contatos, o que pode gerar grafos mais densos, retratando um comportamento homogêneo nas métricas. Com isso, a variação é feita para mostrar como os resultados das métricas podem ser influenciados por tal parâmetro.

A Figura 3 apresenta o gráfico analisando o tamanho da janela versus número de componentes e densidade de arestas do grafo. É possível observar que o valor de 15



(a) Helsinque (b) Roma

Figura 3. Relação entre densidade de arestas e componentes no grafo.

minutos apresenta uma boa proporção de arestas e componentes no grafo. No intuito de melhor caracterizar as variações de comportamentos, neste trabalho optou-se por definir o tamanho de 15 minutos como padrão para o tamanho da janela de tempo.

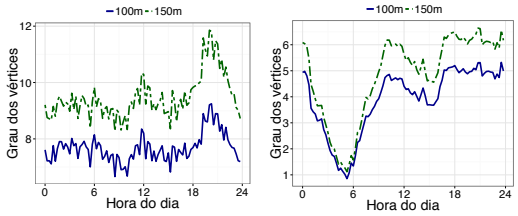
Além disso, a correlação entre algumas métricas foi avaliada. O objetivo desta análise é identificar o quão dependente uma métrica é de outra, e se as métricas apresentam o mesmo comportamento, quando o tráfego varia. Para isso, foi utilizada a Correlação de *Pearson* que mede o grau da relação linear entre duas variáveis, indicando a direção da correlação dentro do intervalo de -1 a 1. Onde o valor é positivo quando uma variável aumenta junto com a outra, ou negativo, quando uma diminui à medida que a outra aumenta. Com isso, o nível da correlação pode ser classificado por três intervalos: fraco (de $0,0$ a $\pm 0,3$), moderado (de $\pm 0,3$ a $\pm 0,7$) e forte (de $\pm 0,7$ a $\pm 1,0$) [1].

4.2. Vértices vs. Vizinhos

Ao avaliar o grau dos vértices, é importante verificar que essa métrica sofre variações ao longo do dia, diante das mudanças na quantidade de veículos em circulação. A Figura 4 apresenta o seu valor médio. Observa-se em Helsinque, Fig. 4-(a), o valor do grau médio permanece constante durante o dia e aumenta ao final, momento de maior densidade de veículos em circulação. Em contrapartida, em Roma (Fig. 4-(b)), observa-se que o valor do grau acompanha a evolução do tráfego. Por serem táxis, veículos que circulam muito, os valores tendem a ser mais semelhantes. Nas duas bases, o aumento do raio da comunicação agregou mais contatos e, consequentemente, o número médio dos graus.

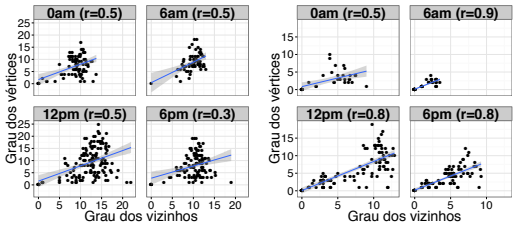
Quando se analisa a correlação entre os graus dos vértices e seus vizinhos, a Figura 5 mostra que os vizinhos dos vértices possuem graus parecidos com os graus dos vértices, uma vez que o Coeficiente de *Pearson* apresentou índices de forte correlação positiva. Isso indica a presença de comportamentos semelhantes dentro das redes, podendo classificá-las como redes assortativas. Redes nas quais os vértices possuem uma forte tendência de encontrar vértices altamente com graus semelhantes conectados entre si.

Quando se verifica a presença de interesses comuns na rede através da busca por comunidades no grafo, a Figura 6, produzida utilizando o algoritmo *Walktrap* [19], mostra que os grafos temporais apresentam comunidades bem definidas. Quando se compara com os grafos aleatórios é possível verificar que a aleatoriedade na geração das arestas quebra as comunidades existentes, deixando comunidades mal definidas e vértices isolados.



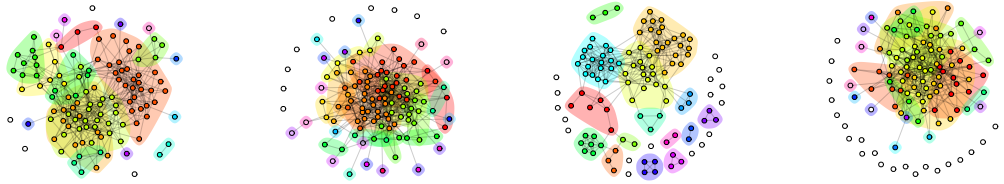
(a) Helsinque (b) Roma

Figura 4. Grau médio dos vértices variado o raio de comunicação.



(a) Helsinque (b) Roma

Figura 5. Correlação entre o grau dos vértices e o grau dos vizinhos.

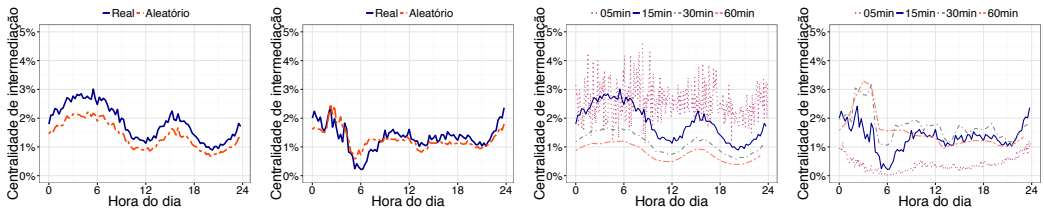


(a) Helsinque real. (b) Helsinque aleatório. (c) Roma real. (d) Roma aleatório.

Figura 6. Comunidades nos grafos temporais e aleatórios.

4.3. Centralidade dos vértices

A centralidade de um veículo define a distância dele em relação a sua distância para os demais veículos do grafo. O quão mais central for o veículo, menor será a sua distância para os demais. A Figura 7-(a) e (b) apresenta os valores para centralidade de intermediação (*betweenness*). Pode-se observar que os valores dos grafos reais são maiores que dos grafos aleatórios, o que comprova que os grafos aleatórios descaracterizam as conexões dos grafos originais. Quando se observa o comportamento destes valores ao variar o tamanho da janela na Figura 7-(c) e (d), nota-se que valores menores para a janela de avaliação apresentam maiores valores de centralidade. Isso ocorre, pois, com janelas menores, a rede apresenta mais desconexões e uma quantidade maior de pequenos componentes.



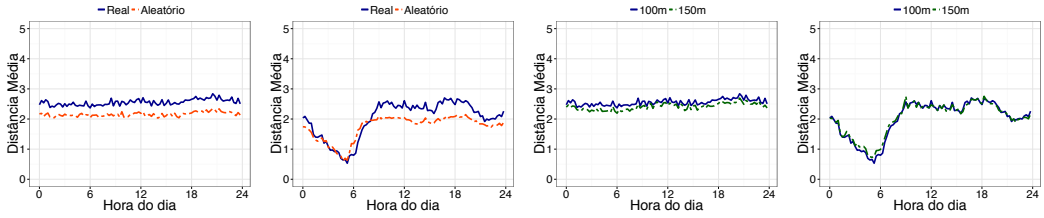
(a) Helsinque (b) Roma (c) Helsinque (d) Roma

Figura 7. Centralidade de intermediação variando o tamanho da janela temporal.

4.4. Mundo pequeno

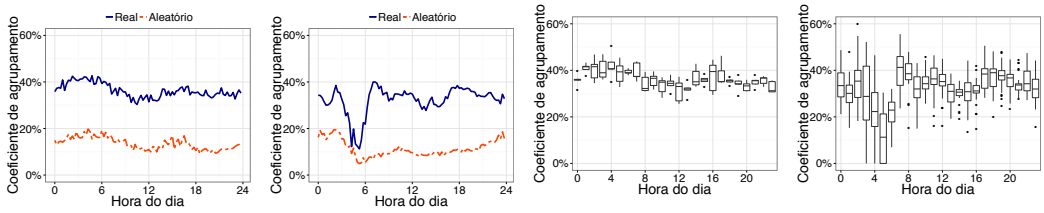
As Figuras 8 e 9 apresentam respectivamente os valores para distância média e coeficiente de agrupamento. Essas métricas retratam a presença de características sociais com rotinas semelhantes e interesses em comum. Baixos valores pra distância e altos valores pra o coeficiente de agrupamento indicam que a rede apresenta o fenômeno *small-world*, uma vez que existem vértices que formam atalhos na rede reduzindo a média da distância global e existe presença de comunidades formando grupos com interesses em comum.

Analisando a distância média, Figuras 8-(a) a (d), observa-se que os grafos aleatórios apresentam valores um pouco menor que as bases reais, devido à ruptura das conexões que ocorre no grafo aleatório. Ao variar o raio de comunicação, observa-se que a existência de mais contatos não impacta no valor da distância média em nenhuma das bases. Nas Figuras 9-(a) e (b) são apresentados os valores do coeficiente de aglomeração para Helsinque e Roma. Pode-se notar que as duas bases apresentam valores maiores quando comparados a grafos aleatórios. Nota-se também que durante as madrugadas os valores são mais altos indicando que os veículos circulam formando grupos provavelmente utilizando as mesmas vias. Pode-se observar também, nas Figuras 9-(c) e (d) a variação dos valores de coeficiente em cada rede. Nota-se que a rede Helsinque, por representar mobilidade de um dia de trabalho apresentam valores maiores do coeficiente quando comparados com a base de Roma. Esta apresenta uma grande variabilidade de valores, alternando entre 20% a 40%.



(a) Helsinque (b) Roma (c) Helsinque (d) Roma

Figura 8. Distância média entre os vértices ao longo do dia.



(a) Helsinque (b) Roma (c) Helsinque (d) Roma

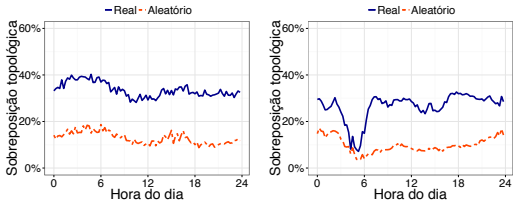
Figura 9. Coeficiente de agrupamento ao longo do dia.

4.5. Semelhança de contatos

A análise da semelhança de contatos pode constatar que ambos os cenários apresentaram rotinas semelhantes e interesses em comum durante o processo de avaliação das métricas

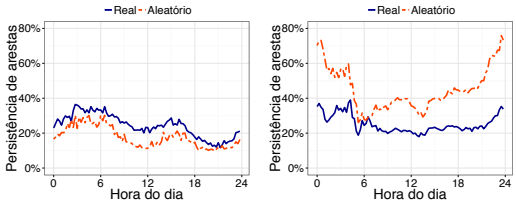
de sobreposição topológica e persistência de arestas, apresentadas nas Figuras 10 e 11. Os resultados obtidos na avaliação da sobreposição topológica nos grafos temporais foram maiores que os grafos aleatórios, da mesma forma que o coeficiente de agrupamento apresentado na Figura 9. Isso indica a existência de grupos de vértices que compartilham interesses em comum. Entretanto ao analisar a repetição dos encontros, Fig. 11-(a) e (b), nota-se que os grafos temporais em Roma apresentaram valores menores que os grafos aleatórios, enquanto que em Helsinque os resultados maiores, indicando que os veículos pessoais repetem mais encontros nos mesmos horários ao longo dos dias, o que não ocorre com táxis. Esses são veículos que retratam o interesse de várias pessoas no mesmo veículo.

Assim, a Tabela 4 indica que a base de Helsinque pode ser classificada como uma rede de amigos, pois ela apresenta características sociais com rotinas semelhantes tanto na sobreposição topológica e quanto na persistência de arestas. Por outro lado, o cenário de Roma pode ser classificado como uma rede de conhecidos, uma vez que os resultados dos grafos temporais não resultaram valores melhores que os grafos aleatórios na avaliação da persistência de arestas.



(a) Helsinque (b) Roma

Figura 10. Sobreposição topológica ao longo do dia.



(a) Helsinque (b) Roma

Figura 11. Repetição de encontros ao longo do dia.

Base de dados	Sobreposição Topológica	Persistência de Arestas	Classe
Helsinque	Social	Social	Amigos
Roma	Social	Aleatório	Conhecidos

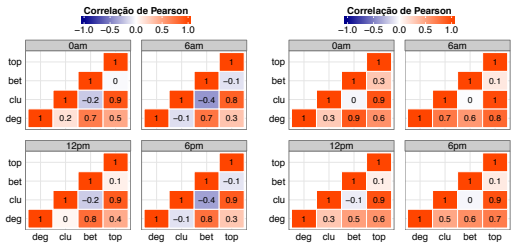
Tabela 4. Classificação das interações social/aleatório baseado na classificação proposta em [24].

4.6. Correlações entre métricas

Com o objetivo de verificar a correlação entre as métricas e assim identificar métricas que possuem o mesmo comportamento, a Figura 12-(a) e (b) apresenta os resultados da correlação de *Pearson* para as métricas sobreposição topológica, coeficiente de aglomeração, grau dos vértices e centralidade de intermediação. Interessante observar que as métricas sobreposição topológica e coeficiente de aglomeração apresentam uma correlação forte, em torno de 0.9 e 1. Ambas são métricas que retratam a presença de interesses comuns entre os vértices, o que explica esse resultado. Entretanto, com as demais métricas a correlação do coeficiente de clusterização e sobreposição topológica é baixa.

Outra correlação pode ser observada entre a centralidade de intermediação e o grau do nó. De fato, quanto maior for o grau de um nó, maior também será a chance de

se ter mais caminhos passando por ele. Comportamento esse explicado pelos valores de 0.7 a 0.9 de correlação entre eles.



(a) Helsinque (b) Roma

Figura 12. Grade de correlações entre as métricas.

5. Aplicações em VSNs

Este trabalho apresenta uma análise de duas bases sociais que descrevem a mobilidade de veículos em um cenário, caracterizando-as como uma rede social veicular com o objetivo de compreender melhor as interações entre os veículos, seus comportamentos e as rotinas dos condutores. Entretanto, esta análise foi executada considerando parâmetros e suposições que caracterizam as interações entre os veículos. Através destas interações foram gerados grafos e computados métricas de redes complexas. Assim, pelos resultados foi possível identificar propriedades e características que indicam fortemente a presença de comportamentos sociais nas redes veiculares.

Muitas são as possibilidades de aplicar essas propriedades e comportamentos no projeto de soluções para rede veicular. No projeto de infraestrutura para a rede, pode-se fazer o uso de métricas como densidade de arestas e distância que ajudam na deposição de ponto de acesso na direção de melhorar a cobertura da rede. Através da densidade de arestas por região é possível identificar locais mais visitados, e pela distância média é possível mensurar a latência na transmissão. Em outra perspectiva, é possível projetar protocolos de comunicação cientes da variação do tráfego e da probabilidade de encontros entre os veículos fazendo uso da métrica grau dos nós [15]. Esta métrica pode auxiliar na definição da probabilidade de retransmitir o dado, o que pode garantir uma melhor cobertura.

Em outra direção, é possível aplicar as métricas para o projeto de aplicações que venham auxiliar os condutores durante o trajeto, tornando a viagem mais segura e agradável. Métricas como coeficiente de agrupamento e sobreposição topológica podem ser usadas para identificar condutores com interesses semelhantes e que visitam os mesmos lugares para o projeto de aplicativos que incentivam caronas e compartilhamento de veículos. Além disso, baseado-se no histórico de visitação será possível criar aplicativos para sugestão de destinos.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma caracterização de duas bases de dados que representam a mobilidade de veículos, buscando encontrar propriedades e características que retratam a interferência do comportamento humano na mobilidade veicular. A base realística representa a mobilidade de veículos pessoais sob o modelo *Working Day Movement Model*,

e a segunda base representa a mobilidade de táxis na cidade de Roma. Esta análise foi realizada considerando os conceitos e métricas de redes complexas e técnicas estatísticas.

Durante a análise das bases, foi possível identificar características semelhantes às redes sociais. Apesar das bases serem de naturezas distintas, onde não há como fazer uma comparação direta entre os resultados, diversas métricas analisadas mostraram a existência de grupos de veículos formando comunidades de pessoas com indícios de rotinas semelhantes e interesses comuns. Estes comportamentos podem ser explicados pela existência de veículos que frequentam os mesmos lugares nos mesmos períodos do dia. Além disso, os resultados mostraram que o cenário de mobilidade realística apresenta características de uma rede de amigos: onde há grupos de pessoas que se encontram com uma frequência. Em contrapartida, o cenário com mobilidade de táxis em Roma apresentou características de uma rede de pessoas conhecidas: onde há grupos de pessoas que compartilham interesses, mas seus encontros não são frequentes.

Em trabalhos futuros, pretende-se avaliar os resultados das interações entre os veículos do ponto de vista espacial, explorando mais as interações entre eles e os locais visitados. Também pretende-se avaliar outras bases com um número maior de veículos em circulação, efetuando filtros e ajustes espaciais e temporais entre os registros, com o objetivo de possibilitar uma comparação de bases de mesma natureza. Além disso, pretende-se incluir análises de estabilidade entre as comunidades para verificar como é o comportamento delas ao longo do tempo, gerando janelas deslizantes na agregação de contatos temporais. Por fim, pode-se estender esse trabalho buscando aplicar o uso das métricas para o projeto de novas soluções para redes veiculares.

Agradecimentos

Os autores agradecem CAPES, CNPq e FAPEMIG pelo apoio financeiro.

Referências

[1] Richard A Becker, John M Chambers, and Allan R Wilks. The new s language. *Pacific Grove, Ca.: Wadsworth & Brooks, 1988*, 1, 1988.

[2] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD data set roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/roma/taxi/, July 2014.

[3] L da F Costa, Francisco A Rodrigues, Gonzalo Travieso, and Paulino Ribeiro Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, 2007.

[4] Felipe D Cunha, Aline Carneiro Vianna, Raquel A F Mini, and Antonio A F Loureiro. Is it possible to find social properties in vehicular networks? In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–6, June 2014.

[5] Felipe D Cunha, Aline Carneiro Vianna, Raquel A F Mini, and Antonio A F Loureiro. How effective is to look at a vehicular network under a social perception? In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, pages 154–159. IEEE, 2013.

[6] Elizabeth M Daly and Mads Haahr. Social network analysis for information flow in disconnected delay-tolerant manets. *Mobile Computing, IEEE Transactions on*, 8(5):606–621, 2009.

[7] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. Working day movement model. In *Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models*, pages 33–40. ACM, 2008.

[8] Tanja Falkowski and Myra Spiliopoulou. Observing dynamics in community structures. In *Symposium on Network Analysis in Natural Sciences and Engineering*, page 15, 2006.

[9] Marco Fiore and Jérôme Härri. The networking shape of vehicular mobility. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 261–272. ACM, 2008.

[10] Theus Hossmann, Thrasyvoulos Spyropoulos, and Franck Legendre. Know thy neighbor: Towards optimal mapping of contacts to social graphs for dtn routing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[11] Norman Lloyd Johnson and Samuel Kotz. Urn models and their application; an approach to modern discrete probability theory, 1977.

[12] Xin Liu, Zhuo Li, Wenzhong Li, Sanglu Lu, Xiaoliang Wang, and Daoxu Chen. Exploring social properties in vehicular ad hoc networks. In *Proceedings of the Fourth Asia-Pacific Symposium on Internetware*, page 24. ACM, 2012.

[13] Nicholas Loulloudes, George Pallis, and Marios D Dikaiakos. The dynamics of vehicular networks in urban environments. *arXiv preprint arXiv:1007.4106*, 2010.

[14] Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[15] Romeu Monteiro, Susana Sargento, Wantanee Viriyasitavat, and Ozan K Tonguz. Improving vanet protocols via network science. *arXiv preprint arXiv:1301.0965*, 2013.

[16] Diala Naboulsi and Marco Fiore. On the instantaneous topology of a large-scale urban vehicular network: the cologne case. In *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*, pages 167–176. ACM, 2013.

[17] Mark E J Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[18] George Pallis, Dimitrios Katsaros, Marios D Dikaiakos, Nicholas Loulloudes, and Leandros Tassiulas. On the structure and evolution of vehicular networks. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009. MASCOTS'09. IEEE International Symposium on*, pages 1–10. IEEE, 2009.

[19] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer, 2005.

[20] Cristiano Rezende, Azzedine Boukerche, Richard W Pazzi, Bruno P S Rocha, and Antonio A F Loureiro. The impact of mobility on mobile ad hoc networks through the perspective of complex networks. *Journal of Parallel and Distributed Computing*, 71(9):1189–1200, 2011.

[21] Francisco A Rodrigues. Caracterização, classificação e análise de redes complexas. *Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos*, 2007.

[22] Gautam S Thakur, Pan Hui, and Ahmed Helmy. The structure and traffic flow anatomy of the planet-scale urban vehicular mobility. *Networking Science*, 3(1-4):13–23, 2013.

[23] Sandesh Uppoor and Marco Fiore. Insights on metropolitan-scale vehicular mobility from a networking perspective. In *Proceedings of the 4th ACM international workshop on Hot topics in planet-scale measurement*, pages 39–44. ACM, 2012.

[24] Pedro O S Vaz de Melo, Aline C Viana, Marco Fiore, Katia Jaffrès-Runser, Frédéric Le Mouël, and Antonio A F Loureiro. Recast: Telling apart social and random relationships in dynamic networks. In *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, pages 327–334. ACM, 2013.

[25] Anna Maria Vegni and Valeria Loscri. A survey on vehicular social networks. *Communications Surveys & Tutorials, IEEE*, 17(4):2397–2419, 2015.

[26] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440–442, 1998.

Trilha Principal do SBRC 2016
Sessão Técnica 2
Replicação de Máquinas de Estado

Especificação de Replicação Máquina de Estados Dinâmica

Eduardo Adilio Pelinson Alchieri[1] **, Alysson Neves Bessani**[2] **, João Sousa**[2]

[1] Departamento de Ciência da Computação, Universidade de Brasília

[2]Faculdade de Ciências, Universidade de Lisboa

Resumo. *A replicação Máquina de Estados (RME), abordagem muito utilizada na implementação de sistemas tolerantes a falhas, consiste em replicar os servidores e coordenar as interações entre os clientes e as réplicas, com o intuito de que as mesmas apresentem a mesma evolução em seus estados. Apesar de alguns sistemas implementarem uma RME com capacidade de reconfiguração, não existe nenhuma especificação a respeito dos aspectos relacionados com a reconfiguração de uma RME. Neste sentido, este artigo especifica as propriedades e as suposições necessárias para uma RME dinâmica, além de propor um protocolo que satisfaz estas propriedades. Além disso, apresenta uma avaliação experimental que facilita o entendimento e mostra a viabilidade prática de uma RME dinâmica.*

Abstract. *State Machine Replication is an approach widely used to implement fault-tolerant systems. The idea behind this approach is to replicate the servers and to coordinate the interactions among clients and servers replicas, making all of these replicas present the same state evolution. Although some systems implement a reconfigurable RME, this approach has not yet been fully specified. In this sense, this paper specifies the properties and the assumptions that are necessary to a reconfigurable RME. Moreover, it presents a protocol that satisfies these properties and an experimental evaluation that shows the practical feasibility of this approach.*

1. Introdução

A Replicação Máquina de Estados (RME) [Schneider 1990, Lamport 1998] é uma abordagem muito utilizada na implementação de sistemas tolerantes a falhas, tanto por parada [Lamport 1998, Schneider 1990] quanto Bizantinas [Castro and Liskov 2002]. Esta abordagem consiste em replicar os servidores e coordenar as interações entre os clientes e estas réplicas, com o intuito de que as mesmas apresentem a mesma evolução em seus estados, garantindo o determinismo de réplicas.

A grande maioria das concretizações existentes para RME (ex.: [Castro and Liskov 2002, Clement et al. 2009, Santos and Schiper 2013]) consideram que o conjunto de réplicas que implementam o sistema nunca sofre alterações. Esta abordagem impossibilita, por exemplo, que durante a execução do sistema novas réplicas sejam adicionadas e/ou réplicas antigas sejam removidas em tempo de execução. No entanto, muitos sistemas necessitam executar estas ações quando querem aumentar ou diminuir o limite de falhas toleradas ou ainda trocar um servidor antigo com uma configuração obsoleta por um servidor atualizado. Para isso, surge a necessidade de *reconfiguração* do sistema.

O sistema SMART [Lorch et al. 2006], que tolera apenas falhas por parada, e o sistema BFT-SMART [Alchieri et al. 2013, Bessani et al. 2014], que suporta falhas tanto por parada quanto Bizantinas, são gratas exceções que implementam uma RME reconfigurável. Lamport *et al.* [Lamport et al. 2010] também discutem, embora superficialmente (nem mesmo apresentam protocolos), aspectos sobre a reconfiguração de uma RME. Um sistema reconfigurável fornece as interfaces necessárias para sua reconfiguração, não se preocupando com as políticas relacionadas com a escolha tanto do momento da reconfiguração quanto da nova configuração a ser implantada no sistema. Os algoritmos que implementam estes sistemas devem prever a possibilidade de entradas e saídas de réplicas. Estas ações de reconfiguração devem ser sincronizadas com as execuções dos protocolos da RME, a fim de preservar a consistência dos estados das réplicas.

Embora existam estas implementações, ainda falta uma especificação formal e precisa a respeito dos aspectos relacionados com a reconfiguração de uma RME. Os trabalhos sobre RME dinâmica não discutem, por exemplo, quanto tempo um servidor precisa permanecer no sistema após ser removido pela execução de uma reconfiguração ou quais são as novas suposições necessárias para o sistema funcionar em um ambiente dinâmico.

Visando preencher esta lacuna, este artigo apresenta as seguintes contribuições: *(1)* especificação de propriedades e de suposições necessárias para uma RME dinâmica, além de uma série de definições relacionadas com o dinamismo do sistema; *(2)* proposta de um protocolo que, considerando as suposições definidas, satisfaz as propriedades especificadas para uma RME dinâmica; e *(3)* implementação e análise experimental do protocolo proposto, o que ajuda a entender o funcionamento de uma RME dinâmica.

2. Modelo de Sistema

Consideramos um sistema distribuído completamente conectado composto pelo conjunto universo de processos U, que é dividido em dois subconjuntos: um conjunto infinito de servidores $\Pi = \{s_1, s_2, ...\}$ e um conjunto infinito de clientes $C = \{c_1, c_2, ...\}$. A chegada dos processos segue o modelo de chegadas infinitas com concorrência desconhecida mas finita [Aguilera 2004]. Desta forma, em cada instante de tempo real t da execução, o número de processos executando alguma ação no sistema é desconhecido mas finito. Contudo, processos podem chegar em qualquer momento (chegadas infinitas) e também passarão a participar das computações executadas no sistema.

Com relação ao modelo de sincronia, consideramos um sistema parcialmente síncrono [Dwork et al. 1988], necessário para garantir a terminação dos protocolos da RME [Schneider 1990, Lamport et al. 1982, Castro and Liskov 2002]. Além disso, os protocolos propostos são genéricos suficientes para serem usados em um sistema que tolera tanto falhas por parada (CFT - *Crash Fault-Tolerant* RME) [Schneider 1990] quanto Bizantinas (BFT - *Byzantine Fault-Tolerant* RME) [Castro and Liskov 2002]. Na especificação da RME dinâmica (Seção 4), são definidos os tamanhos dos quóruns e o número de falhas toleradas pelos protocolos para cada um destes modelos de falhas.

3. Replicação Máquina de Estados

A Replicação Máquina de Estados (RME) consiste em replicar os servidores e coordenar as interações entre os clientes e estas réplicas, com o objetivo de que as mesmas

apresentem a mesma evolução em seus estados. Conceitualmente, o modelo básico de programação de uma RME envolve duas primitivas (Figura 1):

- *invoke(operation)*: utilizada pelos clientes para invocar operações no serviço implementado pela RME.
- *execute(operation)*: implementada pelos servidores replicados, sendo utiliza sempre que uma operação deve ser executada pela RME.

De um modo geral, um cliente utiliza a primitiva *invoke(operation)* para enviar a requisição *operation* para as réplicas, as quais executam a primitiva *execute(operation)* para processar *operation* e retornar a resposta a ser enviada para o cliente.

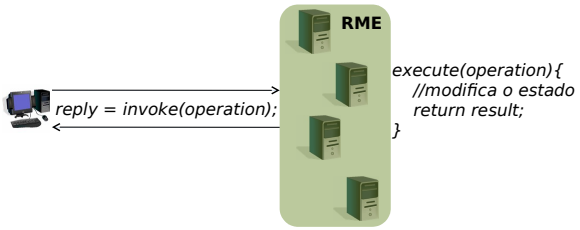


Figura 1. Modelo básico de programação de uma RME.

A implementação destas primitivas deve atender aos seguintes requisitos, que definem o determinismo de réplicas:

1. Estado inicial: todas as réplicas corretas devem iniciar a partir de um mesmo estado.
2. Determinismo: todas as réplicas corretas, que executam uma mesma operação sobre um mesmo estado, realizam a mesma mudança em seus estados e produzem a mesma resposta para o cliente.
3. Coordenação: todas as réplicas corretas executam a mesma sequência de operações.

Para garantir o item *(1.)* basta iniciar todas as réplicas com o mesmo estado (i.e., iniciar todas as variáveis que representam o estado com os mesmos valores nas diversas réplicas), o que é trivial em uma RME estática mas, como veremos mais adiante, requer cuidados especiais em uma RME dinâmica. Para prover o item *(2.)*, é necessário que as operações executadas pelos servidores sejam determinísticas. Finalmente, para que as réplicas executem a mesma sequência de operações afim de garantir o item *(3.)*, é necessária a utilização de um protocolo de difusão atômica [Hadzilacos and Toueg 1994][1].

Assumindo que estes três requisitos sejam satisfeitos, uma implementação de RME deve satisfazer as seguintes propriedades de segurança e terminação:

- Segurança: todas as réplicas corretas executam a mesma sequência de operações.
- Terminação: todas as operações de clientes corretos são executadas.

A propriedade de segurança permite a implementação de consistência forte, conhecida como linearizabilidade (*linearizability*) [Herlihy and Wing 1990]. Já a propriedade de terminação garante que as operações invocadas por clientes corretos são exe-

[1]Difusão atômica, ou difusão com ordem total, consiste em fazer com que todos os processos corretos, membros de um grupo, entreguem todas as mensagens difundidas neste grupo na mesma ordem.

cutadas e acabarão terminando. Os protocolos propostos para RME limitam a quantidade de falhas que o sistema pode apresentar para que estas propriedades sejam atendidas, sendo necessário $n \geq 2f + 1$ (CFT) [Lamport 1998] ou $n \geq 3f + 1$ (BFT) [Castro and Liskov 2002] réplicas para tolerar até f falhas.

4. Replicação Máquina de Estados Dinâmica

Esta seção apresenta algumas definições e propriedades que uma RME dinâmica deve satisfazer, juntamente com as suposições necessárias para sua execução. Posteriormente, são discutidas as etapas para uma reconfiguração.

4.1. Definições e Propriedades de uma RME Dinâmica

As seguintes definições são necessárias na especificação de uma RME dinâmica.

***Updates*.** Definimos um $update = \{+, -\} \times \Pi$, onde a tupla $\langle +, i\rangle$/$\langle -, i\rangle$ representa o pedido de entrada/saída (*join/leave*) do servidor i. O processo de *reconfiguração* considera os *updates* solicitados para definir a nova composição do sistema.

Visões. Uma visão v é composta por um identificador único (representado por $v.id$) e seu *membership* (representado por $v.members$). Por simplicidade, usamos $i \in v$ para simbolizar $i \in v.members$. Duas visões v_1 e v_2 são comparadas através de seus identificadores. Caso $v_1.id < v_2.id$, então v_2 é *mais atual* do que v_1.

Dizemos que uma visão v é *instalada em um servidor correto* caso o mesmo considere v sua *visão atual* e passa a executar operações nesta visão. Neste caso, dizemos que v foi *instalada no sistema*. Em qualquer tempo real t, definimos $V(t)$ como sendo *a visão mais atual* instalada no sistema.

***Bootstrapping*.** Assumimos uma visão inicial não vazia $V(0)$ conhecida por todos os processos do sistema. Durante a inicialização do sistema, cada servidor $i \in V(0)$ recebe a visão inicial $v_0 = \{\langle +, j\rangle : j \in V(0)\}$.

Visões vs. operações de clientes. Em qualquer tempo real t, operações de clientes são executadas apenas em $V(t)$. Quando é requisitado um *join* no sistema para um servidor i, estas operações permanecem desabilitadas em i até a ocorrência de um evento *enable operations*. Após isso, i permanecerá apto a processar e responder a estas operações até que seja requisitado um *leave* do sistema para i, o que acontece com a ocorrência do evento *disable operations*.

Definição 1 (Propriedades de uma RME Dinâmica) *As seguinte propriedades são satisfeitas por uma RME dinâmica:*

◦ ***RME (Segurança)**: Todas as réplicas corretas que executam uma operação possuem o mesmo estado, o qual reflete a execução da mesma sequência de operações.*

◦ ***RME (Terminação)**: Todas operações de clientes corretos são executadas.*

◦ ***Reconfiguração – Join (Segurança)**: Se um servidor correto j instala a visão v tal que $i \in v$, então foi invocada a operação $invoke(\langle +, i\rangle)$ ou i é membro da visão inicial.*

◦ ***Reconfiguração – Leave (Segurança)**: Se um servidor correto j instala a visão v tal que $i \notin v \wedge (\exists v' : i \in v' \wedge v'.id < v.id)$, então foi invocada a operação $invoke(\langle -, i\rangle)$.*

◦ ***Reconfiguração – Join (Terminação)****: O evento enable operations terminará por ocorrer em todo servidor correto i cuja operação $invoke(\langle +, i \rangle)$ foi requisitada.*

◦ ***Reconfiguration – Leave (Terminação)****: O evento disable operations terminará por ocorrer em todo servidor correto i cuja operação $invoke(\langle -, i \rangle)$ foi requisitada.*

Para suportar a dinamicidade do sistema, a propriedade de segurança da RME precisou ser modificada (e de certa forma enfraquecida) para permitir que nem todos os servidores executem a mesma sequência de operações. De fato, não faz sentido que um servidor que entra no sistema seja obrigado a executar todas as operações anteriores já executadas no mesmo. Ao invés disso, modificamos esta propriedade para especificar que o estado deste servidor deve refletir as execuções passadas, o que é suficiente para manter o determinismo de réplicas. Isso é garantido pela transferência de estado quando da instalação de uma nova visão. Por outro lado, a propriedade de terminação da RME não foi alterada e define que todas as operações de clientes corretos são executadas, possibilitando que as mesmas terminem.

Além disso, foram definidas *(a)* duas propriedades de segurança da reconfiguração que garantem que um processo entra e/ou sai do sistema apenas se a operação ($invoke$) correspondente foi executada e *(b)* duas propriedades de terminação da reconfiguração que garantem que as operações são habilitadas e/ou desabilitadas em um servidor, uma vez que a operação ($invoke$) correspondente foi requisitada.

4.2. Suposições para uma RME Dinâmica

Para lidar com a dinamicidade do sistema, as seguintes suposições são necessárias.

Suposição 1 (Limiar de falhas) *Para cada visão v, denotamos $v.f$ como sendo o número de falhas toleradas em v, $v.f \leq \lfloor \frac{|v.members|-1}{2} \rfloor$ (CFT) ou $v.f \leq \lfloor \frac{|v.members|-1}{3} \rfloor$ (BFT).*

Suposição 2 (Tamanho dos quóruns) *Para cada visão v, consideramos quóruns de escrita de tamanho $v.q_e = \lceil \frac{|v.members|+1}{2} \rceil$ (CFT) e $v.q_e = \lceil \frac{2|v.members|+1}{3} \rceil$ (BFT) e quóruns de leitura de tamanho $v.q_l = 1$ (CFT) e $v.q_l = \lceil \frac{|v.members|+2}{3} \rceil$ (BFT).*

Suposição 3 (Saídas graciosas) *Um servidor correto $i \in V(t)$, cuja operação $invoke(\langle -, i \rangle)$ é requisitada no tempo t, permanece no sistema até enviar seu estado para pelo menos um quórum de escrita da visão seguinte na sequência $V(t'), t' > t, i \notin V(t')$.*

Suposição 4 (Reconfigurações finitas) *O número de updates requisitados em uma execução é finito.*

As suposições 1 e 2 são adaptações diretas dos resultados para sistemas estáticos [Lamport 1998, Castro and Liskov 2002]. A Suposição 3 diz que um servidor correto que está deixando o sistema participará do protocolo de reconfiguração. Já a Suposição 4 garante que operações de clientes sempre terminam, i.e., garante que um cliente lento reiniciará sua operação, devido a reconfigurações, um número finito de vezes e, então, completará sua operação[2]. Esta suposição é necessária em todas as propostas de protocolos que envolve clientes acessando um sistema dinâmico [Alchieri et al. 2014, Aguilera et al. 2011, Martin and Alvisi 2004], pois para não utilizar dados obsoletos os mesmos precisam acessar a visão mais atual do sistema.

[2]Apenas o número de *updates* requisitados concorrentemente com uma operação deve ser finito.

4.3. Fases de uma Reconfiguração

Durante a execução de uma RME dinâmica, uma sequência de visões $V_0 \rightarrow ... \rightarrow V_{i-1} \rightarrow V_i \rightarrow V_{i+1} \rightarrow ...$ é instalada para computar os pedidos de entrada (*join*) e de saída (*leave*) do sistema. A dinamicidade do sistema é observada na medida em que novas visões são instaladas. O estado inicial das réplicas de cada visão V_i é definido como o estado final das réplicas da visão anterior na sequência V_{i-1}, garantindo que nada daquilo que foi processado em V_{i-1} seja perdido. A atualização de uma visão V_i, que compreende sua desativação e a instalação da visão seguinte V_{i+1}, ocorre através de um procedimento de reconfiguração do sistema, o qual pode ser dividido nas seguintes fases (Figura 2):

(Fase 1) Determinação da ordem de execução da operação de reconfiguração. A ideia é que as operações de reconfiguração sejam processadas pela RME como uma operação normal, i.e., o protocolo de difusão atômica é executado para definir a ordem de entrega desta operação. Assim, um processo que tenha o privilégio de invocar uma operação de reconfiguração (ex.: um administrador), executa a operação $invoke(\langle +/-, id_1 \rangle, ..., \langle +/-, id_n \rangle)$ passando o conjunto de *updates* (ver Seção 4.1) que deve ser aplicado na visão corrente (V_i), o que resultará na geração da visão seguinte V_{i+1}.

(Fase 2) Execução da operação de reconfiguração. Quando a operação de reconfiguração é executada, os *updates* requisitados são aplicados em V_i para gerar a visão seguinte na sequência V_{i+1} (como esta operação é ordenada, todas as réplicas em V_i a executam no mesmo ponto da execução distribuída, de forma que a mesma visão V_{i+1} é gerada em todas as réplicas). Neste ponto, os processos em V_i param de executar operações (nenhuma outra operação é executada em V_i), desativam V_i, enviam seus estados para os processos em V_{i+1}, enviam a resposta para o invocador da operação de reconfiguração, e deixam o sistema (para aqueles que estiverem saindo).

(Fase 3) Inicialização da nova visão. Por fim, os processos em V_{i+1}, após atualizarem seus estados, instalam V_{i+1} e passam a receber, ordenar e executar operações.

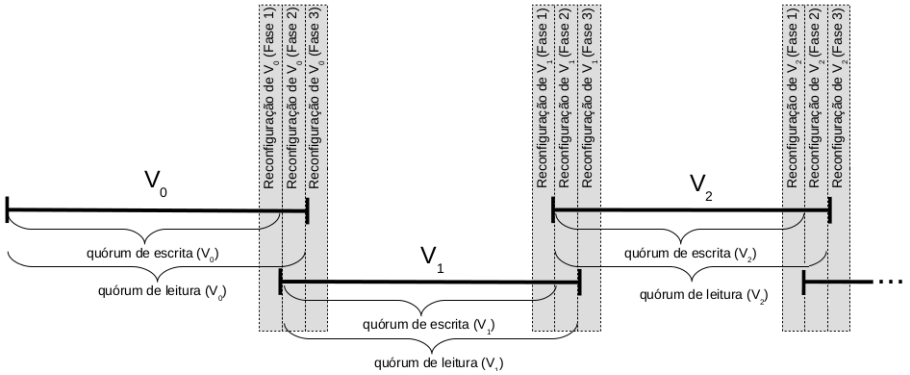


Figura 2. Visões X Reconfigurações.

Como podemos observar na Figura 2, o *ciclo de vida* de uma visão V_i começa com o início da fase 2 da reconfiguração (mesmo que V_i seja instalada apenas na fase 3), quando V_i é gerada através da aplicação de um conjunto de *updates* na visão anterior na sequência V_{i-1}, e termina no final da fase 2 da reconfiguração seguinte, quando V_i é desativada e seu estado é transferido para os servidores da visão seguinte V_{i+1} e a resposta é enviada para o processo que executou a operação de reconfiguração ($invoke$). Para tolerar até f falhas, deve-se aguardar por 1 (CFT) ou $f + 1$ (BFT) respostas iguais para então assumir que esta resposta é correta e finalizar a execução da primitiva *invoke*.

Apesar de serem necessárias $n \geq 2f+1$ (CFT) ou $n \geq 3f+1$ (BFT) réplicas para tolerar até f falhas, nem todas as fases da reconfiguração envolvem todas estas réplicas, mas sim um subconjunto delas chamado de quórum. Os protocolos subjacentes da RME, como o de ordenação, também utilizam a ideia de quóruns para votações. Desta forma, definimos quórums para escrita e para leitura (ver Seção 4.2). Considerando uma visão V_i, um quórum de escrita deve estar disponível deste o início da fase 2 da reconfiguração que a gerou, pois é necessário que as réplicas da visão anterior V_{i-1} atualizem o estado de pelo menos um quórum de escrita de V_i para garantir que o estado inicial de V_i corresponde ao estado final de V_{i-1}, até o final da fase 1 da reconfiguração seguinte que instalará V_{i+1} pois é necessário executar o protocolo de difusão atômica para definir a ordem de execução da operação de reconfiguração. Por outro lado, um quórum de leitura de V_i deve ficar disponível até o final da fase 2 para que a operação de reconfiguração seja executada, i.e., para que *(1)* a visão V_{i+1} seja gerada, *(2)* os servidores de V_{i+1} recebam um número suficiente de mensagens e consigam atualizar seus estados e *(3)* o processo que invocou a operação de reconfiguração receba um número suficiente de respostas e termine.

5. Protocolo para uma Replicação Máquina de Estados Dinâmica

Esta seção descreve o protocolo proposto para concretização de uma RME dinâmica. Para suportar reconfigurações, a ideia principal é ordenar os pedidos de reconfiguração juntamente com as requisições normais de clientes, de modo que os mesmos sejam executados por todas as réplicas no mesmo ponto da execução distribuída. Assim, uma única sequência de visões é instalada no sistema e o estado da aplicação (réplicas) é transferido entre estas visões, fazendo com que uma visão começe a funcionar com o estado da visão anterior na sequência. Já para executar operações normais, os clientes apenas devem se certificar de que estão utilizando a visão mais atual instalada no sistema, garantindo que dados atualizados são acessados.

Note que não desejamos modificar os protocolos subjacentes da RME. Por exemplo, não modificamos os protocolos da RME tradicional que fazem as contagens de respostas e a ordenação. Ao invés disso, utilizamos as primitivas da RME tradicional e incluimos os controles necessários para suportar o dinamismo.

Algoritmo. O Algoritmo 1 apresenta o protocolo utilizado pelos clientes para acesso à RME dinâmica. Basicamente, cada cliente deve armazenar uma variável cv com a visão mais atual conhecida, a qual deve ser incluida em cada requisição enviada para a RME (linha 6): através desta primitiva, a requisição $\langle operation, cv \rangle$ é enviada para todos os servidores em cv, aguardando por respostas $reply$ iguais de $cv.q_l$ (quórum de leitura) servidores em cv para retornar $reply$. Caso a visão do cliente esteja desatualizada, a resposta para sua requisição indicará que sua visão precisa ser atualizada (linhas 7-8), então a requisição é reenviada para a RME (linhas 5-9). Caso contrário, a requisição termina retornando a resposta (linha 10).

Este algoritmo também apresenta os procedimentos para o pedido de entrada (linhas1-2) e de saída (linhas 3-4) de uma réplica do sistema. Geralmente, somente um administrador pode acessar estes procedimentos, principalmente se consideramos um ambiente sujeito a falhas maliciosas. Para simplificar a apresentação dos protocolos, consideramos que apenas um *update* é solicitado em cada requisição, mas o protocolo apresentado pode ser facilmente estendido para processamento de vários *updates* por requisição.

Algoritmo 1 RME dinâmica – $join$, $leave$ e $invoke$ – cliente i.

```
variables: Visão atual do processo i
    cv ← v_0                                              // visão mais atual de i
procedure join(id)
 1) cv ← invoke(⟨+, id⟩)          //executa o invoke solicitando a inclusão (join) da réplica id.
 2) return cv
procedure leave(id)
 3) cv ← invoke(⟨−, id⟩)          //executa o invoke solicitando a remoção (leave) da réplica id.
 4) return cv
procedure invoke(operation)
 5) repeat
 6)    reply ← invoke(⟨operation, cv⟩)                  //requisição é enviada e executada na RME.
 7)    if reply = ⟨need_update_view, v⟩ then      //caso o cliente esteja com a visão desatualizada, ...
 8)       cv ← v                                  //... atualiza a visão e reenvia a requisição para a RME.
 9) until reply = ⟨response, cv⟩                          //caso a visão do cliente estava atualizada, ...
10) return response                                       //... termina retornando a resposta.
```

O Algoritmo 2 apresenta o protocolo executado por cada réplica que suporta a RME dinâmica. Cada servidor também deve armazenar a visão mais atual cv instalada no sistema e, para cada requisição recebida, verificar se a visão do cliente está atualizada para então processar a operação, seja ela uma reconfiguração (linhas 3-15) ou uma operação normal (linha 17). Caso a visão do cliente esteja desatualizada, a resposta para a requisição será a visão atualizada (linhas1-2), possibilitando que o cliente reenvie sua requisição para a nova visão.

No processamento de uma reconfiguração, primeiramente a nova visão é definida (linhas 5-9). Caso a réplica esteja saindo do sistema, suas operações são desabilitadas (linhas 10-11), i.e., nenhuma outra operação é executada pela réplica. Após isso, o estado da aplicação (função $getState$ retorna o estado da réplica) é enviado para uma réplica que esteja entrando no sistema (linhas 12-13) e a visão atualizada é instalada pela réplica (linha 14). A resposta desta operação será a nova visão do sistema (linha 15).

Uma réplica que está entrando no sistema atualiza sua visão e o estado da aplicação (função $setState$ atualiza o estado da réplica) quando receber estas informações de um quórum de leitura (linhas 19-21), i.e., um quórum de leitura enviou o mesmo estado e visão[3]. Quando isso acontece, as operações são habilitadas nesta réplica que passa a executá-las (linhas 1-18). Note que apenas é necessário atualizar o estado das réplicas que estão entrando no sistema, pois as que estavam no sistema já estão atualizadas.

Transferência de Estado. A atualização do estado das réplicas que estão entrando no sistema pode ser fator determinante para o desempenho do protocolo de reconfiguração, pois dependendo do tamanho do estado, um tempo considerável pode ser dispendido nesta tarefa. Tentando diminuir este impacto, o estado pode ser transferido antes ou após a reconfiguração. A transferência antes da reconfiguração apenas alivia o trabalho durante a reconfiguração, pois tudo aquilo que foi modificado desde a transferência deve ser novamente enviado para a réplica. Por outro lado, transferir o estado após a reconfiguração faz com que processos que estejam deixando o sistema precisem aguardar até que o estado das réplicas que entraram seja atualizado, o que ocorrerá apenas após a reconfiguração.

[3]Adicionalmente, pode ser necessário enviar informações dos protocolos subjacentes da RME (ex.: protocolo de ordenação), a fim de que as réplicas sincronizem suas execuções.

Como as operações de clientes apenas serão executadas nas novas réplicas após as mesmas atualizarem seus estados, isso pode causar um atraso na execução das mesmas. No entanto, caso o *membership* da nova visão sofra pequenas alterações, de forma que um quórum de leitura da visão antiga permaneça na nova visão, nenhum atraso será gerado pois existe um número suficiente de réplicas com o estado atualizado aptas a executar e responder às requisições de clientes e, além disso, a reconfiguração será mais rápida pois não ocorre transferência de estado durante a mesma.

Algoritmo 2 RME dinâmica – $execute$ – réplica/servidor i.

variables: Visão atual do processo i
$cv \leftarrow v_0$ // visão mais atual de i

{***Réplicas em cv.***}
upon $execute(\langle operation, v_{req}\rangle)$ //vai executar uma requisição, caso já foram habilitadas na linha 21.
1) **if** $cv \neq v_{req}$ **then** //caso a visão do cliente esteja desatualizada,...
2) **return** $\langle need\_update\_view, cv\rangle$ //... "envia" a visão atualizada para o cliente.
3) **else**
4) **if** $operation = \langle +, id\rangle \vee operation = \langle -, id\rangle$ **then** //caso for uma operações de reconfiguração ...
5) $new\_v.id \leftarrow cv.id + 1$ //... cria a nova visão incrementando seu identificador e ...
6) **if** $operation = \langle +, id\rangle$ **then**
7) $new\_v.members \leftarrow cv.members \cup id$ //... atualiza seu *membership* adicionando ou ...
8) **else**
9) $new\_v.members \leftarrow cv.members \setminus id$ //... removendo o identificador de alguma réplica.
10) **if** $i \notin new\_v$ **then** //caso não esteja na visão atualizada, i.e., saiu do sistema, ...
11) *disable operations* //... desabilita a execução de operações.
12) **if** $operation = \langle +, id\rangle$ **then** //caso uma réplica esteja entrando no sistema, ...
13) $send\langle$STATE-UPDATE, $getState(), new\_v, cv\rangle$ to id //... envia o estado para a mesma.
14) $cv \leftarrow new\_v$ //então, desativa a visão antiga atualizando cv – instala a nova visão.
15) $response \leftarrow cv$ //a resposta será a visão atualizada.
16) **else** //caso for uma operação "normal" de clientes, ...
17) $response \leftarrow execute(operation)$ //executa a operação na réplica, gerando a resposta.
18) **return** $\langle response, v_{req}\rangle$ //"envia" a resposta para o cliente.

{***Réplicas aguardando para entrar no sistema.***}
upon receipt of $\langle$STATE-UPDATE, $state, new\_v, old\_v\rangle$ from $old\_v.q_l$ servers in old_v
19) $cv \leftarrow new\_v$ //quem está entrando instala a visão corrente ...
20) $setState(state)$ //... atualiza o estado da réplica ...
21) *enable operations* //... e habilita a execução de operações.

Lidando com a Dinamicidade do Sistema. Como explicado, clientes e servidores devem ter a informação mais atual sobre a composição do sistema. Este gerenciamento é trivial nos servidores, visto que os mesmos geram e instalam as visões, mantendo-se sempre com a configuração do sistema atualizada. No entanto, manter esta informação atualizada nos clientes é mais difícil e, para isso, os servidores devem informá-los sobre as atualizações na composição do sistema na medida em que operações são requisitadas.

No entanto, um cliente pode permanecer um longo período de tempo sem acessar o sistema, de forma que sua visão não contenha nenhum servidor da visão atual. Neste cenário, não é possível informá-lo que existe uma visão mais atual. Além disso, considerando v a visão atual do sistema, para uma requisição ser ordenada e entregue, a mesma deve atingir pelo menos um servidor correto de v. Consequentemente, um cliente que utiliza uma visão antiga w apenas terá sua requisição ordenada e obterá uma resposta contendo a visão atualizada v, caso $w \cap v$ contenha pelo menos um servidor correto.

Para resolver estes problemas, as visões do sistema podem ser armazenadas também em algum repositório, a partir do qual os clientes podem obtê-las. Para tolerar falhas, as visões podem ser armazenadas em vários repositórios ou em algum serviço tolerante a falhas. Nesta abordagem, sempre que um cliente não tenha sua requisição atendida dentro de um determinado tempo, o mesmo deve verificar no repositório se está utilizando a visão atual do sistema.

5.1. Síntese das Provas de Corretude

Esta seção apresenta um rascunho das provas de que o protocolo apresentado na seção anterior garante as propriedades de uma RME dinâmica (Definição 1).

Lema 1 *Todas as réplicas corretas que executam uma operação possuem o mesmo estado, o qual reflete a execução da mesma sequência de operações.*

Síntese da Prova. Através dos protocolos da RME tradicional [Lamport 1998, Castro and Liskov 2002], dentro de uma mesma visão as operações são executadas seguindo a mesma ordem e, por serem deterministas, geram a mesma alteração no estado das réplicas. Adicionalmente, todas as réplicas da visão inicial começam a execução com o mesmo estado. Desta forma, para provar esta propriedade, basta provar que *(a)* as visões instaladas no sistema formam uma única sequência $V_0 \rightarrow ... \rightarrow V_{i-1} \rightarrow V_i \rightarrow V_{i+1} \rightarrow ...$ e *(b)* o estado final da visão V_i é transferido e usado como estado inicial da visão V_{i+1}. Primeiramente vamos provar *(a)*. Como a execução de uma operação de reconfiguração r para uma visão V_i é ordenada juntamente com as outras operações antes de ser executada, quando da execução de r todos os servidores de V_i possuem o mesmo estado e instalam uma mesma visão seguinte V_{i+1} (linhas 4-14, 19-21, Alg. 2). Nenhuma outra visão diferente de V_{i+1} pode ser gerada para suceder V_i, uma vez que as réplicas que estão saindo do sistema desabilitam suas operações (linha 11, Alg. 2) e as réplicas que ficam no sistema atualizam suas visões para V_{i+1} (linha 14, Alg. 2). Consequentemente, uma única sequência de visões é instalada no sistema. A prova de *(b)* deriva diretamente do fato de que todas as réplicas de V_i possuem o mesmo estado final (quando da execução de r) e enviam (linha 13, Alg. 2) este estado para as réplicas que entram no sistema em V_{i+1} (linhas 19-21, Alg. 2). Consequentemente, o estado inicial de todas as réplicas em V_{i+1} é o estado final das réplicas de V_i. □

Lema 2 *Todas operações de clientes corretos são executadas.*

Síntese da Prova. Considere uma operação o executada por um cliente correto c (linhas 5-10, Alg. 1). Neste caso, o é enviada para todos os processos da visão corrente de c (cv_c), que através das propriedades dos protocolos da RME tradicional [Lamport 1998, Castro and Liskov 2002] acabarão por ordenar e executar o (linhas 1-18, Alg. 2). Desta forma, existem duas possibilidades: *(Caso 1)* cv_c é a visão atual do sistema – a resposta de todos servidores corretos será o resultado da execução de o e cv_c (linhas 17-18. Alg. 2), uma vez que em uma RME as operações são executadas segundo uma ordem, fazendo com que c termine (linhas 9-10, Alg. 1); e *(Caso 2)* cv_c está desatualizada – a resposta de todos servidores corretos será a indicação de que a visão do cliente precisa ser atualizada (linha 2. Alg. 2), o qual a atualiza e volta a enviar a requisição para a RME repetindo este processamento até que a mesma seja executada (linhas 5-9, Alg. 1), consequentemente, como o número de reconfigurações é finito (Suposição 4), cv_c acabará por ficar atualizada e chegamos ao *(Caso 1)*. Como em qualquer um dos casos o é executada, por generalização, todas as operações de clientes corretos são executadas. □

Lema 3 *Se um servidor correto j instala a visão v tal que $i \in v$, então foi invocada a operação $invoke(\langle +, i\rangle)$ ou i é membro da visão inicial.*

Síntese da Prova. Considere que um servidor correto j instala a visão v tal que $i \in v$. Neste caso, i é membro da visão inicial ou j executou a requisição $\langle operation, w\rangle$ (linhas 1-18, Alg. 2) tal que $operation = \langle +, i\rangle$ e $w.id = v.id - 1$. Consequentemente, a operação $invoke(\langle +, i\rangle)$ foi invocada (linhas 1-2, Alg. 1). □

Lema 4 *Se um servidor correto j instala a visão v tal que $i \notin v \wedge (\exists v' : i \in v' \wedge v'.id < v.id)$, então foi invocada a operação $invoke(\langle -, i\rangle)$.*

Síntese da Prova. Considere que um servidor correto j instala a visão v tal que $i \notin v \wedge (\exists v' : i \in v' \wedge v'.id < v.id)$. Neste caso, j executou a requisição $\langle operation, w\rangle$ (linhas 1-18, Alg. 2) tal que $operation = \langle -, i\rangle$ e $w.id = v.id - 1$. Consequentemente, a operação $invoke(\langle -, i\rangle)$ foi invocada (linhas 3-4, Alg. 1). □

Lema 5 *O evento enable operations terminará por ocorrer em todo servidor correto i cuja operação $invoke(\langle +, i\rangle)$ foi requisitada.*

Síntese da Prova. Considere que a operação $invoke(\langle +, i\rangle)$ foi requisitada (linhas 1-2, 5-10, Alg. 1). Pelo Lema 2, os servidores de uma visão v acabarão por executar a requisição $\langle\langle +, i\rangle, v\rangle$ (linhas 1-18, Alg. 2), gerando uma visão $w : w.id = v.id{+}1 \wedge i \in w$. Desta forma, os servidores corretos de v enviam os seus estados e w para i (linha 13, Alg. 2). Pelas suposições 1, 2 e 3, $v.q_l$ servidores corretos enviam seus estados para i, fazendo com que o evento *enable operations* ocorra em i (linhas 19-21, Alg. 2). □

Lema 6 *O evento disable operations terminará por ocorrer em todo servidor correto i cuja operação $invoke(\langle -, i\rangle)$ foi requisitada.*

Síntese da Prova. Considere que a operação $invoke(\langle -, i\rangle)$ foi requisitada (linhas 1-2, 5-10, Alg. 1). Pelo Lema 2, os servidores de uma visão v acabarão por executar a requisição $\langle\langle -, i\rangle, v\rangle$ (linhas 1-18, Alg. 2), gerando uma visão $w : w.id = v.id{+}1 \wedge i \notin w$. Consequentemente, o evento *disable operations* ocorrerá em i (linhas 10-11, Alg. 2). □

6. Experimentos

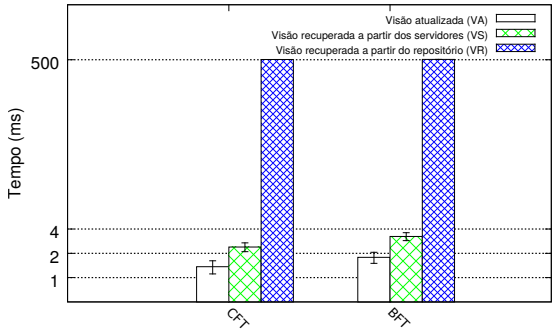
O protocolo proposto foi implementado no BFT-SMaRt [Alchieri et al. 2013, Bessani et al. 2014], que representa a concretização de uma RME desenvolvida na linguagem de programação Java. O BFT-SMaRt conta com protocolos para *checkpoints* e transferência de estados, tornando-se assim uma biblioteca completa para RME. Para atualização do estado das réplicas que entram no sistema, utilizamos o módulo de *checkpoints* e transferência de estado já implementado pela biblioteca [Bessani et al. 2013], de forma que seguimos então a abordagem de atualização de estado após a reconfiguração.

Visando analisar o desempenho da implementação desenvolvida, bem como o comportamento de uma RME dinâmica, alguns experimentos foram realizados no Emulab [White et al. 2002], em um ambiente consistindo por 6 máquinas *d710* (2.4 GHz 64-*bit* Intel Quad Core Xeon E5530, 12GB de RAM e interface de rede gigabit) conectadas a um *switch* de 1Gb. Cada servidor executou em uma máquina separada, enquanto que 10 clientes executaram em outra máquina. O ambiente de *software* utilizado foi o SO Ubuntu 12.04 LTS 64-*bit* com *kernel* 3.2.0-56 e máquina virtual Java de 64 *bits* versão 1.7.0_75.

A aplicação utilizada nos experimentos foi uma lista encadeada que armazena inteiros, sendo que a mesma foi inicializada com $100k$ entradas em cada réplica. Utilizamos nos clientes apenas a operação *Integer get(int index)*, que retorna o elemento da posição indicada[4]. Os clientes selecionam o parâmetro desta operação de forma aleatória, entre 0 e $100k - 1$, seguindo uma distribuição uniforme.

Os experimentos realizados foram: (1) *Latência* – primeiramente foi determinada a latência para a execução de operações considerando diferentes cenários para a visão utilizada pelos clientes; e (2) *Throughput* – também determinamos o *throughput* atingido nos servidores durante uma execução com diferentes reconfigurações que adicionam e removem servidores do sistema. Todos os experimentos tiveram uma fase de *warm-up*.

Latência. A Figura 3 apresenta a latência média, para o sistema configurado tanto como CFT (visão atual cv: $|cv| = 3 \wedge cv.f \leq 1$) quanto BFT (visão atual cv: $|cv| = 4 \wedge cv.f \leq 1$), para os três casos possíveis: *(a)* VA – a visão do cliente está atualizada; *(b)* VS – a visão do cliente está desatualizada e contém servidores que ainda estão no sistema, então a visão é recuperada a partir destes servidores; e *(c)* VR – a visão do cliente está desatualizada e não contém servidores que estão no sistema, então após um *timeout* (configurado em 500 ms) a visão é recuperada a partir do repositório. Neste experimento, utilizamos como repositório um diretório compartilhado na rede por todos os processos. A latência foi medida em um dos clientes (sempre o mesmo) e os valores apresentados representam a média de 1000 execuções, excluindo-se 10% dos valores com maior desvio. Em nenhuma das execuções o desvio padrão ultrapassou 0.32 ms.



		Atualização	Execução	Total
CFT	VA	0	1.364	1.364
	VS	1.252	1.144	2.396
	VR	501.456	1.867	503.323
BFT	VA	0	1.782	1.782
	VS	1.648	1.590	3.238
	VR	501.604	2.701	504.305

Figura 3. Latência (tempos em ms) quando o cliente possui a visão atualizada (VA) ou a recupera a partir dos servidores (VS) ou do repositório (VR).

Quando o cliente possui a visão atualizada o tempo apresentado reflete a execução normal de uma operação no sistema. Para o caso da visão ser recuperada a partir dos servidores, o tempo apresentado reflete a execução de duas operações: uma para atualizar a visão (a resposta da primeira tentativa de execução é a visão atualizada – Algorithm 2) e outra para efetivamente executar a operação. Já para o terceiro caso, o tempo apresentado contempla: o envio da requisição para os servidores; a espera pelo *timeout*; a leitura e atualização da visão a partir do repositório; e a execução da operação. A Figura 3 (direita) apresenta uma tabela com os tempos gastos para atualizar a visão e para efetivamente executar a operação após a visão estar atualizada. Como podemos perceber, no caso *(b)* a latência é aproximadamente o dobro do caso *(a)*, enquanto que no caso *(c)* a latência sofre grande impacto do *timeout* para aguardo das respostas.

[4]Utilizamos apenas esta operação porque o objetivo destes experimentos não é analizar a aplicação lista em si, mas o comportamento de uma RME dinâmica em diversos cenários.

Throughput. A Figura 4 mostra o *throughput* obtido pelo sistema, configurado como CFT, considerando eventos de entradas e saídas de réplicas em uma execução de 300 segundos. O *throughput* foi medido em um dos servidores (o líder do protocolo de ordenação [Bessani et al. 2014]). Inicialmente, o sistema é suportado por 3 réplicas, i.e., $v_0 = \{0, 1, 2\}$. As réplicas 3 e 4 são inseridas no sistema após 60 e 120 segundos, respectivamente, permanecendo no sistema por 120 segundos, i.e., a réplica 3 deixou o sistema no segundo 180 e a réplica 4 no segundo 240. Desta forma, o sistema apresentou as seguintes visões de acordo com o intervalo de tempo: $0 - 60$ ($v_0 = \{0, 1, 2\}$, $v_0.f \leq 1$); $60-120$ ($v_1 = \{0, 1, 2, 3\}$, $v_1.f \leq 1$); $120-180$ ($v_2 = \{0, 1, 2, 3, 4\}$, $v_2.f \leq 2$); $180-240$ ($v_3 = \{0, 1, 2, 4\}$, $v_3.f \leq 1$); e $240 - 300$ ($v_4 = \{0, 1, 2\} = v_0$, $v_4.f \leq 1$).

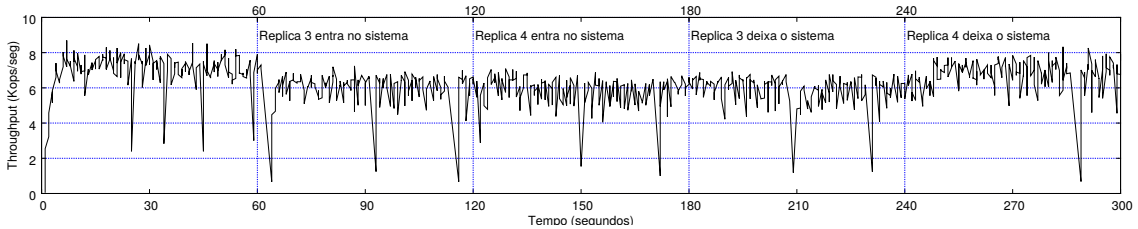


Figura 4. Throughput.

Conforme os clientes iniciam suas execuções, o *throughput* vai aumentando até que todos estejam em execução, o que acontece por volta dos 5 segundos. A partir do segundo 60, o desempenho do sistema sofreu uma pequena queda devido ao aumento do número de réplicas e do tamanho dos quóruns de escrita (de 2 para 3 réplicas). Após isso, o *throughput* não teve variação significativa com a entrada da réplica 4 ou com a saída da réplica 3, pois o tamanho dos quóruns se manteve constante em 3. Finalmente, com a saída da réplica 4 o sistema voltou a ter apenas 3 réplicas (e quóruns de escrita de tamanho 2), apresentando um desempenho semelhante ao inicial. Outro ponto a destacar é que de tempos em tempos o sistema apresentou uma queda brusca de desempenho devido à execução do protocolo de *checkpoints* [Bessani et al. 2013, Bessani et al. 2014].

7. Conclusões

Este artigo apresentou uma proposta para especificação de uma RME dinâmica, onde o conjunto de réplicas que suporta o sistema pode sofrer alterações em tempo de execução. Definimos três fases para o processo de reconfiguração e identificamos as quantidades de réplicas que devem permanecer no sistema em cada fase. Além disso, definimos as propriedades para uma RME dinâmica e apresentamos um protocolo para satisfazê-las. O protocolo proposto foi implementado e os resultados dos experimentos mostram o comportamento prático de uma RME com suporte a reconfigurações.

Referências

Aguilera, M. (2004). A pleasant stroll through the land of infinitely many creatures. *SIGACT News*, 35(2):36–59.

Aguilera, M. K., Keidar, I., Malkhi, D., and Shraer, A. (2011). Dynamic atomic storage without consensus. *JACM*, 58:7:1–7:32.

Alchieri, E. A. P., Bessani, A. N., and da Silva Fraga, J. (2013). Replicação máquina de estados dinâmica. In *Anais do XIV Workshop de Teste e Tolerância a Falhas*.

Alchieri, E. A. P., Bessani, A. N., Greve, F., and da Silva Fraga, J. (2014). Reconfiguração modular de sistemas de quóruns. In *Anais do XXXII Simpósio Brasileiro de Redes de Computadores*.

Bessani, A., Santos, M., Felix, J. a., Neves, N., and Correia, M. (2013). On the efficiency of durable state machine replication. In *Proceedings of the USENIX Annual Technical Conference*.

Bessani, A., Sousa, J., and Alchieri, E. (2014). State machine replication for the masses with BFT-SMaRt. In *Proc. of the International Conference on Dependable Systems and Networks*.

Castro, M. and Liskov, B. (2002). Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

Clement, A., Kapritsos, M., Lee, S., Wang, Y., Alvisi, L., Dahlin, M., and Riché, T. (2009). UpRight cluster services. In *Proc. of the ACM SOSP'09*.

Dwork, C., Lynch, N. A., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of ACM*, 35(2):288–322.

Hadzilacos, V. and Toueg, S. (1994). A modular approach to the specification and implementation of fault-tolerant broadcasts. Technical report, Department of Computer Science, Cornell.

Herlihy, M. and Wing, J. M. (1990). Linearizability: A correctness condition for concurrent objects. *ACM Trans. on Programing Languages and Systems*, 12(3):463–492.

Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine generals problem. *ACM Transactions on Programing Languages and Systems*, 4(3):382–401.

Lamport, L. (1998). The part-time parliament. *ACM Transactions Computer Systems*, 16(2):133–169.

Lamport, L., Malkhi, D., and Zhou, L. (2010). Reconfiguring a state machine. *SIGACT News*, 41:63–73.

Lorch, J. R., Adya, A., Bolosky, W. J., Chaiken, R., Douceur, J. R., and Howell, J. (2006). The smart way to migrate replicated stateful services. In *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*.

Martin, J.-P. and Alvisi, L. (2004). A framework for dynamic Byzantine storage. In *Proceedings of the International Conference on Dependable Systems and Networks*.

Santos, N. and Schiper, A. (2013). Achieving high-throughput State Machine Replication in multi-core systems. In *Proc. of the IEEE ICDCS'13*.

Schneider, F. B. (1990). Implementing fault-tolerant service using the state machine aproach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., and Joglekar, A. (2002). An Integrated Experimental Environment for Distributed Systems and Networks. In *Proc. of 5th Symp. on Operating Systems Design and Implementations*.

Replicação Tolerante a Faltas Eficiente em Bancos de Dados Orientados a Grafos

Ray Willy Neiheiser[1], Lau Cheuk Lung[1], Aldelir Fernando Luiz[2], Hylson Vescovi Netto[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina

[2]*Campus* Blumenau – Instituto Federal de Educação, Ciência e Tecnologia Catarinense

{neiheiser.r, hylson.vescovi}@posgrad.ufsc.br, lau.lung@ufsc.br, aldelir.luiz@blumenau.ifc.edu.br

Abstract. *With the growth of social networks and the increase of highly linked data, graph technologies like distributed graph databases gained importance. Replication, widely used in the field of traditional databases is also implemented by various graph database solutions. Unfortunately, existing approaches for other database systems offer weak performance and scalability. We therefore consider deferred update replication using atomic broadcast to offer a scalable fault-tolerant solution for distributed graph databases. Our contribution is the adaption of the deferred update algorithm, offering highly scalable replication with performance advantage of when compared to traditional replication approaches.*

Resumo. *Com o crescimento das redes sociais e o aumento no volume de dados altamente conectados, a tecnologia banco de dados baseada em grafos tem ganhado notória importância. A replicação de dados, que é uma técnica amplamente adotada no âmbito de bancos de dados tradicionais, é também passível de implementação em soluções de bancos de dados baseados em grafos. Porém, as abordagens existentes noutros ambientes de banco de dados oferecem pouca escalabilidade e desempenho. Neste sentido, este trabalho introduz a técnica de atualização tardia (i.e., deferred update) no contexto de sistemas bancos de dados orientados a grafos. A solução proposta consiste numa adaptação do algoritmo clássico desta abordagem, de modo a possibilitar uma replicação bastante escalável, e com desempenho superior quando comparado à replicação tradicional.*

1. Introdução

Os avanços tecnológicos ocorridos nas últimas décadas nas áreas de computação e comunicação têm estimulado o surgimento de novas classes de aplicações que, cada vez mais, têm produzido um imenso volume de dados – principalmente aqueles baseados na Web, como por exemplo, as redes sociais. Estas aplicações demandam por uma gestão eficiente em termos dos dados manipulados, o que não é facilmente provido por um sistema de banco de dados tradicional, tal como o relacional. Tal eficiência é requerida para que seja possível obter escalabilidade e desempenho aceitáveis quanto ao uso destas aplicações. Por conseguinte, tal demanda tem culminado na proposição de novos modelos, paradigmas e tecnologias para a concepção de bancos de dados capazes de prover as necessidades requeridas por estas aplicações [Stonebraker 2010].

É lícito salientar que, ao longo dos anos os sistemas de gerenciamento de banco de dados relacionais (SGBDR) têm se intensificado como um padrão de *facto* como a solução

mais adequada/ideal para a persistência e recuperação de dados em sistemas computacionais de propósito geral [Stonebraker et al. 2007] – o que se deve principalmente pela sua maturidade [Codd 1970]. Todavia, a dinamicidade e a complexidade inerentes à concepção das aplicações mais modernas, associada ao volume de dados produzido/utilizado por elas, podem ser vistos como novos requisitos de aplicações aos sistemas de bancos de dados, o que por sua vez expõem novos desafios para os Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDRs) tradicionais.

Embora os benefícios quanto ao uso dos SGBDRs sejam indiscutíveis para aplicações de propósito geral, aplicações modernas baseadas em modelos de dados não relacionais, e portanto, diferentes daqueles suportados pelos SGBDRs, são passíveis de sofrerem consequências devido à pouca eficiência do SGBDRs, no que tange a realização de operações sobre dados não normalizados/estruturados [Hellerstein et al. 2007]. Em face deste fato, uma nova tecnologia de banco de dados, denominada NoSQL (*Not Only SQL*) [Cattell 2011] surge no intuito de suprir as necessidades de aplicações que processam e armazenam grandes volumes de dados (p. ex.: dados complexos, semi-estruturados ou não estruturados). E por assim dizer, a tecnologia trazida pelo NoSQL visa, sobretudo, melhorar questões relacionadas a escalabilidade, disponibilidade e desempenho em termos do armazenamento e recuperação de dados.

As principais características que diferenciam os bancos de dados NoSQL dos relacionais tradicionais são: (*i*) o relaxamento das propriedades ACID [Haerder and Reuter 1983], uma vez que eles são norteados pelo princípio conhecido por BASE (*Basically*, *Available*, *Soft State*, *Eventually Consistent*) [Pritchett 2008], e; (*ii*) a possibilidade de se utilizar modelos de dados distintos, de acordo com a necessidade das aplicações (p. ex.: chave-valor, orientado a documentos, orientado a colunas e baseado em grafos). Deste modo, as categorias de bancos de dados NoSQL existentes proveem uma flexibilidade em termos da representações de dados, além de dispensar o rigor necessário à definição dos *schemas*, tal como ocorre no modelo relacional [Stonebraker 2010]. Ademais, cabe informar que a tecnologia de banco de dados NoSQL tem sido amplamente adotada por organizações como Facebook, Amazon e Google, Twitter e LinkedIn.

No que diz respeito ao modelo de dados baseado em grafos, tal estrutura de dados tem sido empregada na modelagem de um grande número de aplicações que visam a solução de problemas práticos e reais (p. ex.: mineração de dados nas redes sociais). Alguns destes problemas são, inclusive, resolvidos através do processamento distribuídos do grafo – um exemplo concreto é o Google *Pregel* [Malewicz et al. 2010]. No intuito de prover um ambiente adequado para o armazenamento de dados de aplicações modeladas por grafos é que surgem os bancos de dados baseados em grafos, como uma alternativa aos sistemas de bancos de dados tradicionais, isto é, os relacionais. Note que o armazenamento de grafos consiste num sistema que contém uma sequência de nodos e suas relações que, quando combinadas, dão origem a um grafo [Robinson et al. 2015].

Por outro lado, embora o armazenamento de grafos seja essencialmente construído a partir de uma estrutura de dados simples do tipo nodo-relação-nodo, há uma complexidade inerente à manutenção de alguns atributos requeridos e preconizados pelo movimento NoSQL [Leavitt 2010, Cattell 2011], como é o caso da escalabilidade. No tocante a este atributo, pode-se dizer que o satisfazimento do mesmo em bancos de dados NoSQL baseados em grafos não é algo trivial, o que se deve principalmente pela ligação estreita existente em cada nodo de um grafo. Note que os dados podem ser replicados em diversos servidores

para prover melhorias de desempenho em termos de operações de leitura. Porém, a complexidade associada a operações de escritas de dados em múltiplos servidores e de consultas que se estendem por vários nodos do grafo, podem culminar na inviabilização do uso deste tipo de sistema.

Neste contexto, este artigo apresenta a proposição de um protocolo de replicação baseado na clássica técnica conhecida por *Deferred Update* [Kemme and Alonso 2000], para bancos de dados NoSQL baseados em grafos. Ao nosso conhecimento, trata-se da primeira iniciativa no sentido de especificar/desenvolver um protocolo de replicação para este tipo de ambiente, o que, portanto, consiste numa contribuição em termos de algorítmica distribuída tolerante a faltas, e bancos de dados distribuídos. Na mesma perspectiva, a principal contribuição trazida pelo trabalho consiste na especificação de um protocolo de replicação para bancos de dados NoSQL baseados em grafos.

E por assim dizer, o restante do artigo está organizado da seguinte maneira. A Seção 2 faz uma breve apresentação da tecnologia de banco de dados baseada em grafos; na Seção 3 é descrita a técnica de replicação *deferred update*; a Seção 4 descreve, em detalhes, a especificação do protocolo proposto, e; na Seção 5 é apresentada uma avaliação de desempenho e discussão dos resultados. Por fim, a Seção 6 conclui o trabalho.

2. Tecnologia de Bancos de Dados Baseados em Grafos

De um modo geral, o modelo de dados baseado em grafos contém três elementos básicos, os quais são: (*i*) os nodos, isto é, os vértices do grafo; (*ii*) os relacionamentos, representado pelas arestas, e; (*iii*) as propriedades, que denotam os atributos dos nodos e relacionamentos. Desta maneira, um banco de dados pode ser representado como um multidígrafo rotulado [Rabuske 1992], em que cada par de nodos pode ser conectado por uma ou mais arestas. O armazenamento baseado em grafo consiste numa 3-tupla, onde cada campo desta tem a finalidade de armazenar apenas um dos elementos básicos do grafo, isto é, o(s) nodo(s), o(s) relacionamentos e a(s) propriedade(s). A Figura 1 ilustra de maneira simplificada, a estrutura/organização de um sistema de armazenamento baseado em grafo.

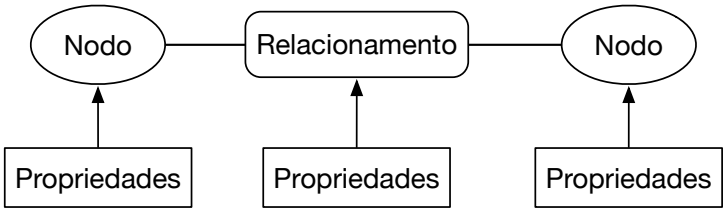


Figura 1. Exemplo de representação do modelo de dados baseado em grafo.

A partir da compreensão do modelo de dados em questão, um banco de dados baseado em grafos pode ser entendido como uma coleção de dados denotada não por um *schema* de dados de dados tradicional, mas por estruturas de dados especificadas a partir de nodos, relacionamentos e propriedades, sob a designação de representar os dados daquela coleção [Robinson et al. 2015] (vide Figura 2). Neste sentido, os nodos são usados para representar entidades como pessoas, negócios, etc., isto é, especificações similares àquelas que visam representar algum objeto de aplicação. Enquanto que as propriedades são usadas para descrever tanto os nodos como os relacionamentos. Por outro lado, os relacionamentos tem a finalidade de ligar (ou relacionar) um nodo a outro, ou a alguma propriedade.

Assim, um sistema de gerência de banco de dados (SGBD) baseado em grafo – tal como o SBGD relacional – é uma componente de software designada a criar, recuperar, atualizar e excluir dados sob a representação de grafos. Todavia, é digno de nota que nem todos os ambientes de bancos de dados baseados em grafos efetuam o armazenamento subjacente de dados de maneira nativa, isto é, na forma de um grafo. Em geral, os sistemas que não proveem o armazenamento nativo em grafos, serializam os dados mapeados no grafo numa representação adequada para os modelos relacional ou de objetos [Robinson et al. 2015]. Alguns exemplos de SGBDs baseados em grafos comerciais suportam armazenamento nativo e merecem destaque: Neo4J, InfoGrid e HyperGraphDB.

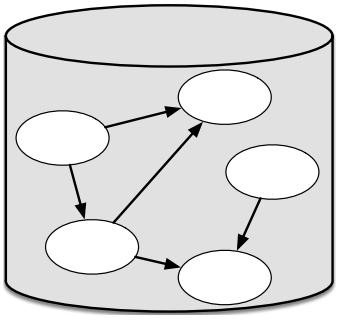


Figura 2. Ilustração de um banco de dados baseado em grafos.

É lícito salientar que as vantagens quanto à utilização do modelo de dados baseado em grafos são mais evidentes quando as aplicações demandam por consultas mais complexas. Neste caso, a opção por um modelo tradicional como o relacional, provavelmente incorrerá num custo demasiadamente grande para a aplicação, já que inúmeras operações de junção serão necessárias para fornecer os dados. Para tais aplicações, os bancos de dados orientados a grafos são uma ótima alternativa, pois são otimizados para efetuar a recuperação de dados em ambientes onde predominam interligações entre os mesmos.

3. Replicação de Dados Baseada na Técnica *Deferred Update*

A literatura descreve a técnica *Deferred Update* como uma das soluções mais promissoras para a replicação de bancos de dados [Kemme and Alonso 2010], num ambiente onde as manipulações de dados ocorrem por meio de transações. Para tanto, a estratégia desta abordagem envolve um conjunto de servidores (ou réplicas), que mantém cópias de um mesmo conjunto de dados (i.e., um banco de dados). A ideia central desta técnica consiste em efetuar a execução de todas as operações de uma transação inicialmente numa única réplica do banco de dados.

Assim, transações que apenas efetuam leituras (i.e., aquelas que não alteram o estado do banco de dados) podem ser validadas localmente naquela réplica em que foram executadas. De outro modo, para transações que alteram o estado do banco de dados (i.e., de escrita), ao término da execução das operações que as compõem, elas devem ser certificadas sobre o ambiente replicado de banco de dados, a fim de verificar se a mesma é passível de validação (i.e., *commit*). Uma vez que uma transação tem êxito em sua certificação, as atualizações por ela efetuadas são propagadas ao ambiente replicado de banco de dados e, após o processamento pelas demais réplicas, a transação é então validada e a partir daí toma efeito no ambiente replicado de banco de dados.

Um aspecto que é digno de nota é que as atualizações podem ser propagadas de maneira consistente às réplicas do ambiente de banco de dados, a partir do uso de uma infi-

nidade de protocolos de sincronização [Pedone et al. 2003]. Tal aspecto, inclusive, tem impelido quanto à adoção da técnica em questão por diversos protocolos de replicação de bancos de dados, e em diferentes contextos [Pedone et al. 2003, Patiño-Martínez et al. 2000, Amir and Tutu 2002]. Ademais, é importante salientar que o uso da mesma tem demonstrado um desempenho bastante aceitável em situações reais/práticas [Pedone et al. 2003, Patiño-Martínez et al. 2000, Amir and Tutu 2002].

Não obstante as benéfices ora elencadas, a respeito da técnica de replicação em lide, outro aspecto que também corrobora para o uso extensivo de tal abordagem em ambientes reais, reside no fato de que nestes ambientes há o predomínio de transações do tipo somente-leitura [Bernstein and Newcomer 2009]. Tal situação permite obter um equilíbrio acerca da carga de trabalho executada pelas réplicas, de modo que elas podem executar as operações de diferentes transações de maneira completamente independente umas das outras; embora ainda seja requerida a sincronização das réplicas para o caso de transações de atualização (i.e., o envio das operações para processamento pelas réplicas). Todavia, em se tratando das escritas, uma otimização bastante recorrente quanto ao uso da técnica é que a maiorias das implementações encapsula todas as operações numa única mensagem. Este encapsulamento permite obter ganhos de escala, bem como evitar sobrecarga do sistema subjacente de comunicação (i.e., quando comparado às técnicas que efetuam a propagação operação a operação).

4. Visão Geral da Solução Proposta

Nesta seção se apresentam os principais aspectos que amparam a solução ora proposta. Neste sentido, conforme já mencionado a solução proposta consiste num protocolo para a replicação de dados em bancos de dados baseados em grafos, a partir da técnica já conhecida, denominada *deferred update*. A escolha da estratégia de replicação se dá por algumas razões como a boa escalabilidade verificada em diversas implementações de tal estratégia de replicação de dados [Patiño-Martínez et al. 2000, Amir and Tutu 2002, Pedone et al. 2003, Kemme and Alonso 2010].

Um aspecto interessante acerca da técnica em questão, é que quando comparada com as abordagens mais clássicas de replicação, tais como a **replicação de máquinas de estados** [Schneider 1990] e a **replicação primário-backup** [Budhiraja et al. 1993]. Por exemplo, a replicação de máquinas de estados não permite um aumento da vazão (i.e., *throughput*) do sistema quando se adicionam novas réplicas – ao contrário do que ocorre com o uso do *deferred update* –, isso porque cada transação é executada por todas as réplicas. Por outro lado, a replicação primário-backup executa a transação sobre a réplica primária, e então propaga as alterações para processamento pelas réplicas de backup. Neste caso, a vazão do sistema é limitada pela capacidade de processamento da réplica primária, e não pelo número de réplicas.

É importante notar que a replicação baseada no *deferred update* apresenta melhor escalabilidade que as demais mencionadas, justamente porque todas as réplicas podem atuar como **primárias**, uma vez que as transações sempre são executadas localmente por alguma réplicas e então propagadas para as demais. Este é o caso do protocolo proposto, conforme ilustrado pela Figura 3. Note que o cliente dá início a uma transação quando submete uma operação para alguma das réplicas (passo 1) – a réplica 2, no caso da Figura 3. Esta réplica é designada como líder da transação, e portanto, tem a função de executar todas as operações submetidas pelo cliente no âmbito daquela transação. No passo 2, ao receber a operação e

processá-la, a réplica líder envia a resposta da operação ao cliente. Esta fase do protocolo conhecida como fase de **execução** perdura até o momento em que o cliente deseja validar sua transação (i.e., efetuar o *commit*) – o que é realizado no passo 3.

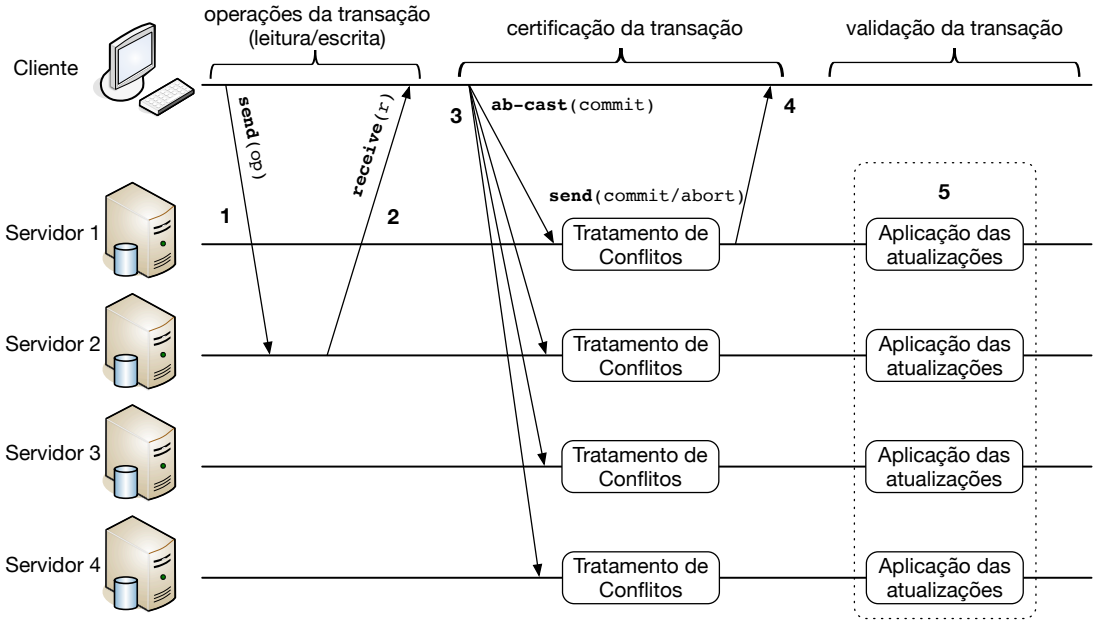


Figura 3. Dinâmica de funcionamento do protocolo.

Quando o cliente conclui a execução da transação, ele envia um pedido de validação (i.e., uma mensagem ***commit***) por meio de difusão atômica (passo 3), o que implica que todas as réplicas terminarão por entregar tal mensagem ao protocolo de replicação. Assim, ao entregar a mensagem em questão, as réplicas dão início a execução de um procedimento para realizar a certificação da transação. Este procedimento verifica se as alterações efetuadas pelas transação estão em consonância com o critério de **serialização** – i.e., se a transação pode ser serializada após a última transação validada e estável. Note que a certificação é efetuada com base na verificação da existência de conflitos entre a transação em processo de validação, em relação as transações já validadas. Uma vez que as réplicas concluem a fase de certificação, iniciada pela entrega da mensagem ***commit***, elas respondem ao cliente acerca do resultado final de transação, o qual será ***commit*** ou ***abort***. E finalmente, no passo 5, após terem enviado suas respostas ao cliente, as réplicas aplicam as atualizações em suas bases de dados locais, a fim de integrar o efeito da transação ao estado do banco de dados. Maiores detalhes a respeito da especificação formal do protocolo serão vistas na Seção 4.2.

4.1. Modelo de Sistema

O ambiente de sistema admitido é composto por um universo de processos $\mathcal{U}$, sendo este dividido em dois subconjuntos. O subconjunto $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ denota as réplicas (ou servidores) do ambiente replicado de banco de dados. Por outro lado, o subconjunto $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ representa os processos dos clientes que interagem com o sistema de banco de dados. O subconjunto $\mathcal{S}$ tem cardinalidade $|\mathcal{S}| \geq 2f + 1$, enquanto que o subconjunto $\mathcal{C}$ pode conter um número arbitrário (não infinito) de processos. Os processos são designados de acordo com seus respectivos comportamentos, isto é, (*i*) um processo é correto se ele segue sua especificação de não falha durante a execução do sistema, ou; (*ii*) é

faltoso, caso contrário. É assumido que até f réplicas dentre as $|\mathcal{S}| \geq 2f + 1$ podem exibir comportamento faltoso, e portanto, parar de executar prematuramente[1]. Em relação aos processos que implementam os clientes, não há um limite para a ocorrência de faltas nestes.

Os processos em $\mathcal{U}$ não compartilham memória, de modo que eles se comunicam apenas por passagem de mensagens. À vista disso, as comunicações ocorrem por meio de primitivas dos tipos **um-para-um** e **um-para-muitos**, sendo que as primitivas *send* e *receive* são empregadas no primeiro caso; e para o segundo caso considera-se o uso de difusão atômica (i.e., com ordem total), cujo envio e entrega de mensagens são denotados ao logo do texto pelas primitivas *A-broadcast* e *A-deliver*, respectivamente. A especificação da primitiva de difusão atômica adotada é baseada no protocolo *Ring-Paxos* [Marandi et al. 2010]. Os canais de comunicação empregados em todos os tipos de comunicações são confiáveis e autenticados [Basu et al. 1996]. Isto implica que, se emissor e receptor são corretos, a mensagem é: (*i*) recebida numa comunicação um-para-um e; (*ii*) entregue numa comunicação um-para-muitos.

Em relação à base de dados, é assumido que cada processo em $\mathcal{S}$ mantém uma cópia completa da mesma, de modo a constituir o ambiente replicado de banco de dados. O sistema de banco de dados adotado é um ambiente nativo de banco de dados baseado em grafo $\mathcal{D} = \{d_i, d_{i+1}, ..., d_n\}$, tal que $i \geq 1 \wedge n < \infty$. Os itens de dados d_i compostos por nodos $\mathcal{N} = \{n_i, n_{i+1}, ..., n_j\}$, relacionamentos $\mathcal{R} = \{r_i, r_{i+1}, ..., r_k\}$ e suas respectivas propriedades. Um relacionamento é sempre orientado e liga exatamente dois nodos n_i e n_j; porém, é importante notar que um relacionamento r_k que liga o nodo n_i com o nodo n_j é, portanto, uma relação diferente daquela que representa uma ligação do nodo n_j com o nodo n_i. É também lícito salientar que um nodo n_i pode ter ou não relacionamentos.

As interações dos clientes com o sistema de banco de dados se dá por meio de transações. O modelo admite o uso de um esquema de travas (i.e., *locks*) durante o processamento de transações, a fim de permitir a execução consistente das operações de leitura e de escrita realizadas no contexto da transação. Ao término da execução das operações, a transação é concluída pela sua validação (*commit*) – que torna permanente as alterações efetuadas pela transações – ou anulação (*abort*) – que descarta todo o efeito da transação. Cada nodo n_i e relacionamento r_j é identificado por um ID único, ou então, pela combinação única de suas propriedades. Ademais, cada transação mantém os conjuntos designados por *read-set* (rs), *write-set* (ws), *delete-set* (ds) e *create-set* (cs), cujos elementos consistem na descrição dos nodos e relacionamentos referenciados pela transação. E finalmente, o critério de consistência adotado é aquele baseado na **serialização**, isto é, aquele cujo efeito da execução concorrente de um conjunto de transações é equivalente ao da execução sequencial do mesmo conjunto de transações (i.e., uma após a outra).

Por fim, a considerar o modelo de interação (i.e., das hipóteses temporais), assumimos o modelo baseado em sincronismo terminal (*eventually synchronous system*) [Dwork et al. 1988] em razão das características realistas presentes em tal modelo – é assíncrono em grande parte do tempo, porém, durante períodos de estabilidade os tempos são limitados mas desconhecidos.

4.2. Base Algorítmica do Protocolo

Nesta seção se apresenta a formalização dos algoritmos que compõem o protocolo proposto. No intuito de facilitar a compreensão por parte do leitor, a especificação do protocolo está

[1]Não são admitidas faltas bizantinas, de modo que os processos falham apenas por parada.

dividida em duas partes, isto é, a parte que trata dos clientes e a parte que trata das réplicas (vide Figuras 1 e 2, respectivamente).

Inicialmente é realizada a declaração das variáveis mantidas pelos clientes para cada transação por eles executada (linhas 1 a 8, Algoritmo 1). Quando um cliente deseja iniciar uma transação, ele o faz por meio do procedimento $begin(t)$, em que t denota o id local da transação atribuído pelo cliente. Tal procedimento simplesmente inicializa as variáveis que serão utilizadas no decurso da transação, RS, WS, CS e DS, que denotam os *read-set*, *write-set*, *create-set* e *delete-set*, respectivamente (linhas 9 a 12, Algoritmo 1).

Uma vez que a transação é iniciada pelo cliente, ele procede com as operações que irão compor a unidade lógica transacional (p. ex.: leituras e/ou escritas) – vide diagrama da Figura 3. Para executar uma operação de leitura o cliente invoca a função $read$ (linhas 19 a 32 – Algoritmo 1). De outro modo, para operações de escrita o cliente invoca a função $write$, especifica nas linhas 33 a 47 do Algoritmo 1. No caso de uma operação de leitura, ao chamar a função $read$ o cliente fornece como argumentos o identificador da transação t, o id do snapshot no qual a transação irá realizar as operação (s_{id}) e o identificador único uid do nodo ou relacionamento sobre o qual a operação será realizada. É importante salientar que o identificador único é obtido por uma composição de dados, de acordo com o tipo de objeto de banco de dados: (*i*) os nodos são identificados por uma composição formada por seus rótulos e suas propriedades; (*ii*) os relacionamentos são identificados pela composição dos rótulos e propriedades dos nodos dos quais faz parte, pelas propriedades do próprio relacionamento, e pelo tipo de relacionamento.

Ao requisitar uma operação de leitura, se tal operação for a primeira executada no âmbito daquela transação, o cliente envia uma mensagem contendo os parâmetros recebidos pela operação $read$ à replica escolhida como líder da transação, e aguarda (*i*) até o recebimento de uma resposta daquela réplica (linhas 19 e 20 do Algoritmo 1), ou; (*ii*) o esgotamento de um temporizador interno. Ao receber a resposta na linha 21, o *snapshot* é inicializado com o que fora recebido na mensagem enviada pela réplica líder (linhas 21 e 22, Algoritmo 1), e o valor v recebido como resultado da réplica é retornado ao cliente como resultado da operação. Por outro lado, se aquela não for a primeira operação da transação, o algoritmo verifica se o item de dados solicitado para leitura (uid) já foi referenciado/manipulado pela transação e está presente nos conjuntos de escrita e criação (linhas 25 e 27, respectivamente) – pela combinação do uid. Neste caso, o item pode ser retornado diretamente dos valores locais do cliente, sem necessidade de interação com a réplica líder. De outro modo, se o item de dados não foi até então manipulado/referenciado pela transação, o cliente envia uma mensagem à replica líder, e aguarda pelo recebimento do resultado da operação (linhas 29 a 32, Algoritmo 1). É digno de nota que, por razões de simplificação, os temporizadores não constam nas especificações dos algoritmos. Entretanto, quando um temporizador é esgotado, uma exceção é lançada ao cliente. Neste caso, o cliente pode reenviar a operação ou então reiniciar a transação numa outra réplica líder.

Ao contrário do que ocorre com as operações de leitura, as operações de escrita não são enviadas à replica líder. Ao observar a especificação da função $write$ (linhas 33 a 47 – Algoritmo 1), se pode verificar que todas as possíveis operações de escrita (i.e., *create*, *update* e *delete*) são efetuadas diretamente sobre os conjuntos locais mantidos pelo cliente (CS, WS e DS). Neste caso, o que ocorre é apenas a manutenção do valor v fornecido como argumento da operação de escrita, ao respectivo conjunto nos termos da operação requisitada (linhas 33, 35, 40 e 46 do Algoritmo 1). Nos mesmos moldes da função $read$,

Algoritmo 1 Tarefa executada pelos clientes.

Declaração de Variáveis:

1: $t.RS = \perp$ /* *ReadSet* */
2: $t.WS = \perp$ /* *WriteSet* */
3: $t.CS = \perp$ /* *CreateSet* */
4: $t.DS = \perp$ /* *DeleteSet* */
5: $s_i = \perp$ /* id do servidor/réplica */
6: $c_i = \perp$ /* id do cliente */
7: $s_{id} = \perp$ /* id do *snapshot* */
8: $it = \perp$ /* id do *snapshot* */

procedure $begin(t)$ /* Procedimento invocado para iniciar a transação */
9: $t.RS \leftarrow \emptyset$ /* inicialização do read-set */
10: $t.WS \leftarrow \emptyset$ /* inicialização do write-set */
11: $t.CS \leftarrow \emptyset$ /* inicialização do create-set */
12: $t.DS \leftarrow \emptyset$ /* inicialização do delete-set */

function $commit(t, s_{id})$
13: **if** $t.CS = \emptyset \wedge t.WS = \emptyset \wedge t.DS = \emptyset$ **then**
14: **return** `committed`
15: **else**
16: *A-broadcast*$(c_i, \langle \texttt{commit}, t, s_{id} \rangle)$ to $\forall s_i \in \mathcal{S}$
17: **wait until** $receive(\langle t, \texttt{outcome} \rangle)$ from some $s_i \in \mathcal{S}$
18: **return** `outcome`

function $read(t, s_{id}, uid)$ /* Operação de leitura */
19: **if** $it = \perp$ **then**
20: $send(c_i, \langle \texttt{read}, t, uid, s_{id} \rangle)$ to s_i
21: **wait until** $receive(\langle uid, v, rs_{id} \rangle)$ from s_i
22: $s_{id} \leftarrow rs_{id}$ /* id *snapshot* como a primeira leitura */
23: **return** v
24: **else**
25: **if** $uid \in t.CS$ **then**
26: **return** $t.CS(uid)$ /* retorna o conteúdo do create-set */
27: **else if** $uid \in t.WS$ **then**
28: **return** $t.WS(uid)$ /* retorna o conteúdo do write-set */
29: **else**
30: $send(c_i, \langle \texttt{read}, t, uid, s_{id} \rangle)$ to s_i
31: **wait until** $receive(\langle uid, v, rs_{id} \rangle)$ from s_i
32: **return** v

function $write(op, t, s_{id}, uid, v)$ /* Operação de escrita */
33: **if** $op = \texttt{create}$ **then**
34: $t.CS \leftarrow t.CS \cup \langle uid, v \rangle$
35: **else if** $op = \texttt{update}$ **then**
36: **if** $uid \in t.CS$ **then**
37: $t.CS \leftarrow t.CS \cup \langle uid, v \rangle$
38: **else**
39: $t.WS \leftarrow t.WS \cup \langle uid, v \rangle$
40: **else if** $op = \texttt{delete}$ **then**
41: **if** $uid \in t.WS$ **then**
42: $t.WS \leftarrow t.WS \setminus \langle uid, * \rangle$
43: $t.DS \leftarrow t.DS \cup \langle uid, v \rangle$
44: **else if** $uid \in t.CS$ **then**
45: $t.CS \leftarrow t.CS \setminus \langle uid, * \rangle$
46: **else**
47: $t.DS \leftarrow t.DS \cup \langle uid, v \rangle$

o argumento t fornecido para a função $write$ denota a transação para a qual a operação está sendo requisitada, op é a operação a ser executada, s_{id} corresponde ao *snapshot* do cliente, e uid dizem respeitos ao item de dados e o valor a ser manipulado sobre o item de dados.

Por fim, quando não houver mais operações para serem executadas numa transação, o passo seguinte consiste na validação da transação, isto é, a tentativa de tornar permanente o efeito produzido por ela sobre o estado do banco de dados. No protocolo proposto isto é feito por meio de uma chamada para a função $commit$ (linhas 13 a 18 – Algoritmo 1). Assim, quando a validação para uma transação é solicitada pelo cliente, o algoritmo analisa os conjuntos mantidos no lado cliente durante o decurso da transação. Se for constatado que todos eles estão vazios – o que indica que a transação era de somente-leitura –, a transação é validada imediatamente, pois não nesta condição nenhuma alteração foi realizada sobre os itens de dados do banco de dados. Do contrário, isto é, se algum dos conjuntos não for vazio, o cliente envia por meio de difusão atômica uma mensagem `commit` à todas as réplicas do ambiente de banco de dados, e aguarda até o recebimento da primeira mensagem `outcome` enviada por alguma réplica. Note que, como é pressuposto que não ocorrem faltas bizantinas, apenas uma mensagem é o suficiente para atestar o resultado do pedido de validação processado pelas réplicas. No caso, como as réplicas são inicializadas num mesmo estado e as validações das transações são entregues às réplicas na mesma ordem – devido ao uso da difusão atômica –, o resultado após o processamento dos pedidos de validação é o mesmo em todas das réplicas [Schneider 1990].

Em se tratando da especificação do código das réplicas, o mesmo é formalizado a partir do Algoritmo 2. Note que as interações dos clientes durante a fase de execução da transação, quando apropriado, ocorrem diretamente com a réplica líder da respectiva transação (linha 29, Algoritmo 2). No caso, quando uma réplica recebe uma operação de leitura de algum cliente, ela imediatamente faz uma chamada à função $read$ especificada para a réplica, passando como argumento todos os dados recebidos na mensagem (linhas 6 a 13 do Algoritmo 2). Por sua vez, A função $read$ inicializa o *snapshot* que será usado para as operações daquela transação – se tal operação é a primeira daquela transação –, e então procede com a recuperação do valor v para o item de dados uid contido na operação, a partir de seu *snapshot* local. Em seguida, a réplica envia ao cliente o resultado da operação, isto é, o valor v (linha 13, Algoritmo 2).

Outrossim, quando o protocolo de difusão atômica subjacente entrega uma mensagem `commit` para o protocolo de replicação proposto (linha 30 – Algoritmo 2), todas as réplicas iniciam o procedimento de validação e certificação da transação, o que ocorre através da invocação à função $commit$. Observe que, como a mensagem foi enviada por meio de difusão atômica, isto implica que todas as réplicas não faltosas terminarão por entregar tal mensagem, numa mesma ordem. Então, ao adentrar na função $commit$ (linhas 14 a 23 do Algoritmo 2), o primeiro passo consiste na verificação quanto à existência de conflitos entre a transação em processo de validação, em relação às transações já validadas que não precedem a transação em questão – isto é, cuja validação ocorreu no decurso da transação em lide.

A verificação de conflitos, que é realizada pela função $conflict\_handler$ ocorre conforme descrito a seguir (linhas 24 a 28 do Algoritmo 2). Para cada transação já validada, cujo *snapshot id* é maior que o *snapshot id* da transação que acabara de solicitar a validação, verifica-se se há alguma intersecção entre as escritas das transações já validadas com as leituras efetuadas pela transação em processo de validação. A interseção indica que a transação

Algoritmo 2 Tarefa executada pelas réplicas (servidores).

Declaração de Variáveis:

1: $gid = 0$ /* inicialização do *snapshot* global */
2: $cws\langle it, ws\rangle = \emptyset$ /* *WriteSet* validado */
3: $ctx\langle it, v\rangle = \emptyset$ /* transações validadas */
4: $ltx\langle it, v\rangle = \emptyset$ /* transações locais */
5: $l_{id} = \perp$ /* id local da transação */

function $read(t, s_{id}, uid)$

6: $l_{id} \leftarrow \perp$
7: **if** $s_{id} = \perp$ **then**
8: $\quad l_{id} \leftarrow gid$
9: $\quad ltx \leftarrow t$
10: **else**
11: $\quad l_{id} \leftarrow s_{id}$
12: $v \leftarrow$ `retrieve`(uid, l_{id})
13: $send(s_i, \langle uid, v, l_{id}\rangle)$ to c_i

function $commit(t, s_{id})$

14: $outcome =$ `conflict_handler`(t, s_{id})
15: **if** $outcome =$ `commit` **then**
16: $\quad cws \leftarrow cws \cup \langle s_{id}, t.WS\rangle$
17: $\quad ctx \leftarrow ctx \cup \langle s_{id}, t\rangle$
18: $send(s_i, \langle$`outcome`$\rangle)$ to c_i
19: **for** $t_i \in ltx$ **do**
20: $\quad$ **if** $t_i.RS \cap ctx[0]$ **then**
21: $\quad\quad$ `abort`(t_i)
22: $\quad$ **else**
23: $\quad\quad$ `execute`$(ctx[0])$

function $conflict\_handler(t, s_{id})$

24: **for** $t_i \in ltx : it > s_{id}$ **do**
25: $\quad$ **if** $\neg((t_i.WS \cup t_i.DS) \cap t.RS)$ **then**
26: $\quad\quad$ **return** `commit`
27: $\quad$ **else**
28: $\quad\quad$ **return** `abort`

Tarefa principal:

upon $receive(\langle$`read`$, t, uid, s_{id}\rangle)$ from c_i
29: **call** `read`(t, s_{id}, uid)
upon $A\text{-}deliver(\langle$`commit`$, t, s_{id}\rangle)$
30: **call** `commit`(t, s_{id})

que acabara de solicitar sua validação efetuou uma leitura de um dado que foi alterado em seu decurso, o que denota uma condição de leitura "suja", e portanto, viola o critério de serialização. Se tal condição não é verificada na linha 25 do Algoritmo 2, a transação pode ser serializada no banco de dados, após a última transação validada (linha 26 – Algoritmo 2); ou, do contrário, a transação será anulada (linhas 27 e 28, Algoritmo 2).

5. Aspectos de Implementação e Avaliação

No intuito de validar o protocolo proposto e realizar uma prova de conceito, foi desenvolvido um protótipo do mesmo, o qual foi implementado sobre o SGBD baseado em grafos Neo4J. Para tanto, os testes foram executados em 4 máquinas Intel-core i7-3770k quad-cores, hyper threading com 4GB RAM. O sistema operacional utilizado nos experimentos foi o Linux Debian 7.4 Wheeze 64 bits, kernel 3.2.54-2. O algoritmo foi implementado na linguagem Java, tendo sido executado sob a JVM 1.8.0_31. Para os experimentos, o

Neo4j (`neo4j.com`) foi configurado em modo de alta disponibilidade, para uma melhor comparação com o protocolo proposto baseado no *deferred update*. Ademais, a opção pela tecnologia do Neo4j se deu em razão do mesmo realizar escritas apenas em um único servidor principal.

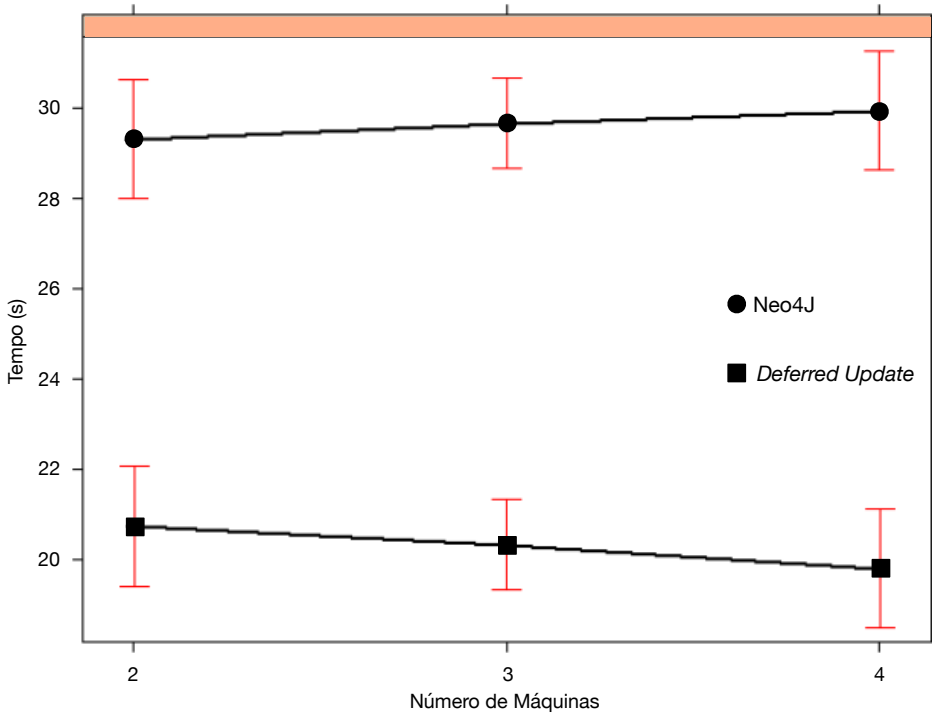


Figura 4. Quatro processos servidores, em duas, três e quatro máquinas físicas.

Foram utilizados bancos de dados inicialmente vazios, e uma função foi projetada no cliente, a fim de enviar 32 transações para os servidores, criando um número aleatório de nós e relacionamentos entre os nós. Foram também criadas operações de atualização dos nós e dos relacionamentos. Durante a execução das transações, podem ocorrer conflitos entre transações concorrentes, que serão resolvidos pelos algoritmos do Neo4J e da solução proposta. Após a execução dos testes, os bancos de dados serão comparados e verificados quanto à consistência. Dois experimentos foram conduzidos para investigar a diferença entre o Neo4J e o protocolo proposto.

O primeiro experimento executou quatro processos servidores sobre duas, três e quatro máquinas físicas. Em cada uma destas três configurações, 10 replicações do experimento foram realizadas, a fim de alcançar resultados estatisticamente relevantes. A seguir, quatro clientes acessaram os quatro processos servidores e exibiram como saída o tempo despendido, assim que os clientes finalizavam a operação. A Figura 4 mostra resultados do Neo4J e do protocolo baseado no *deferred update*. A solução proposta apresenta um desempenho significativamente melhor do que o Neo4J em modo de alta disponibilidade. Neo4J apresenta resultados inferiores quando distribuído em mais servidores, enquanto o algoritmo proposto escala com sucesso, considerando o número de servidores.

O segundo experimento executou de três a oito processos servidores em quatro máquinas físicas. Analogamente ao experimento anterior, de três a oito clientes acessaram os processos servidores em cada configuração, que foi executada dez vezes. Os resultados deste segundo experimento são exibidos na Figura 5. O algoritmo *deferred update* possui

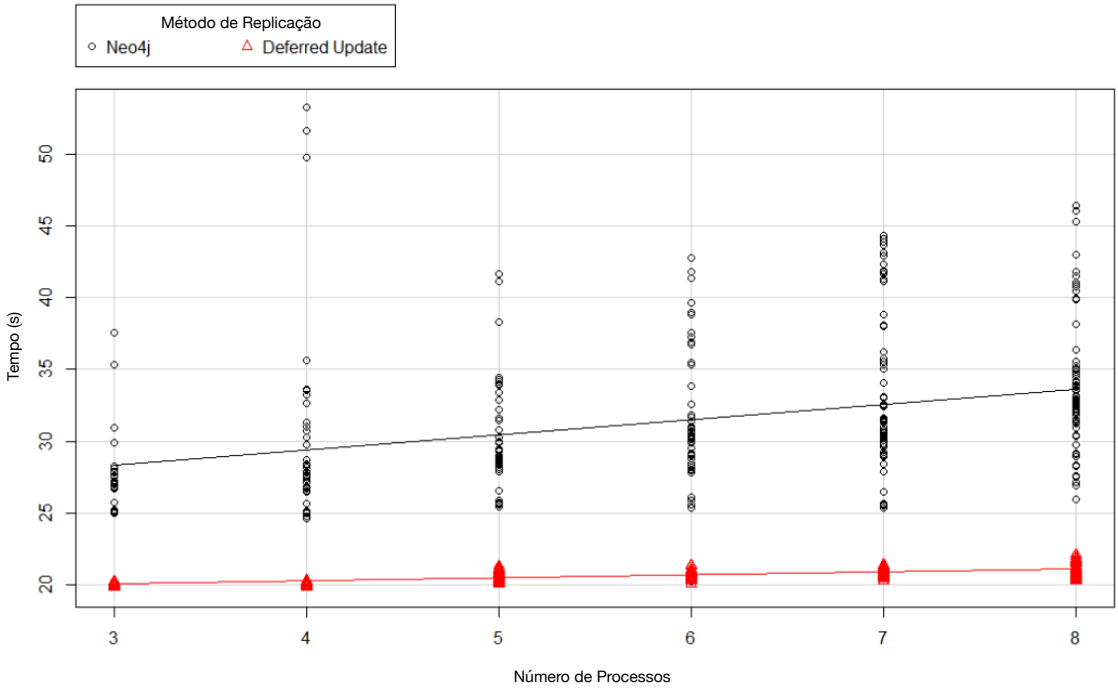


Figura 5. Três a oito processos servidores, em quatro máquinas físicas.

desempenho representado pela linha inferior, enquanto o Neo4j em modo de alta disponibilidade está representado na linha superior. Da mesma maneira que no experimento anterior, o algoritmo proposto demonstrou desempenho superior ao Neo4J. Além disso, a inclinação das linhas mostra que a solução proposta possui uma escalabilidade superior ao Neo4J, em relação ao número de réplicas.

6. Conclusões

O algoritmo apresentado neste artigo, isto é, *deferred update* para bancos de dados baseados em grafos, apresentou desempenho superior à solução desenvolvida para a tecnologia predominante no mercado, o Neo4J. Como possibilidades de expansão das investigações deste trabalho pode-se acessar e utilizar o mecanismo de indexação do Neo4J ou conduzir experimentos em um número maior de réplicas. É digno de nota que a técnica *deferred update* também possui algumas desvantagens. Por exemplo, o algoritmo *deferred update* requer que todas as réplicas contenham o banco de dados inteiro. Embora, na atualidade espaço em disco não seja propriamente um problema, transações de maior duração podem sofrer por inanição. Da mesma maneira, transações longas podem requerer bloqueios globais, o que, portanto, deve ser evitado.

Ademais, a solução proposta pode ser vista como um ponto de partida para que, desenvolvedores de SGBDs baseados em grafos possam implementar soluções escaláveis e ao mesmo tempo, tolerante a faltas em seus respectivos sistemas. Como trabalho futuro, pretende-se estender o protocolo proposto para tolerar também faltas bizantinas.

Agradecimentos

Trabalho parcialmente financiado pela FAPESC/IFC, processo/projeto Nº00001905/2015.

Referências

Amir, Y. and Tutu, C. (2002). From total order to database replication. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 494–503.

Basu, A., Charron-Bost, B., and Toueg, S. (1996). Simulating reliable links with unreliable links in the presence of process crashes. In *Proceedings of the 10th International Workshop on Distributed Algorithms*, pages 105–122.

Bernstein, P. and Newcomer, E. (2009). *Principles of Transaction Processing: for the systems professional*. Morgan Kaufmann Publishers Inc., Burlington, MA, USA, 2 edition.

Budhiraja, N., Marzullo, K., Schneider, F. B., and Toueg, S. (1993). The primary-backup approach. In Mullender, S., editor, *Distributed systems (2nd Ed.)*, pages 199–216. Addison-Wesley Publishing Co., New York, NY, USA.

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *SIGMOD Record*, 39(4):12–27.

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387.

Dwork, C., Lynch, N. A., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of ACM*, 35(2):288–322.

Haerder, T. and Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4):287–317.

Hellerstein, J. M., Stonebraker, M., and Hamilton, J. (2007). Architecture of a Database System. *Foundations and Trends in Databases*, 1(2):141–259.

Kemme, B. and Alonso, G. (2000). A new approach to developing and implementing eager database replication protocols. *ACM Transactions on Database Systems*, 25(3):333–379.

Kemme, B. and Alonso, G. (2010). Database replication: A tale of research across communities. *The Proceedings of the VLDB Endowment*, 3(1):5–12.

Leavitt, N. (2010). Will NoSQL databases live up to their promise? *Computer*, 43(2):12–14.

Malewicz, G., Austern, M. H., Bik, A. J. C., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G. (2010). Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 135–146.

Marandi, P. J., Primi, M., Schiper, N., and Pedone, F. (2010). Ring paxos: A high-throughput atomic broadcast protocol. In *Proceedings of the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 527–536.

Patiño-Martínez, M., Jiménez-Peris, R., Kemme, B., and Alonso, G. (2000). Scalable replication in database clusters. In *Proceedings of the 14th International Conference on Distributed Computing*, pages 315–329.

Pedone, F., Guerraoui, R., and Schiper, A. (2003). The database state machine approach. *Distributed and Parallel Databases*, 14(1):71–98.

Pritchett, D. (2008). BASE: An ACID Alternative. *ACM Queue*, 6(3):48–55.

Rabuske, M. A. (1992). *Introdução à Teoria dos Grafos*. Editora da UFSC, Florianópolis, SC, Brasil.

Robinson, I., Webber, J., and Eifrem, E. (2015). *Graph Databases: New Opportunities for Connected Data*. O'Reilly Media, Inc., 2 edition.

Schneider, F. B. (1990). Implementing fault-tolerant service using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM*, 53(4):10–11.

Stonebraker, M., Madden, S., Abadi, D. J., Harizopoulos, S., Hachem, N., and Helland, P. (2007). The end of an architectural era: (it's time for a complete rewrite). In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 1150–1160.

Replicação de Máquinas de Estado em *containers* no Kubernetes: uma Proposta de Integração

Hylson Netto[1,2], Lau Cheuk Lung[1] , Miguel Correia[3], Aldelir Fernando Luiz[2]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina

[2]Campus Blumenau – Instituto Federal de Educação, Ciência e Tecnologia Catarinense

[3]INESC-ID – Instituto Superior Técnico – Universidade de Lisboa – Portugal

hylson.vescovi@posgrad.ufsc.br, aldelir.luiz@blumenau.ifc.edu.br,
lau.lung@ufsc.br, miguel.p.correia@tecnico.ulisboa.pt

Resumo. *A virtualização de computadores permitiu um provisionamento dinâmico de recursos nos data centers e um consequente modelo de cobrança de serviços de acordo com utilização. A virtualização em nível de sistema por meio dos containers tornou mais dinâmica essa capacidade de provisionamento. Em aplicações que requerem garantias mais restritivas de acordo e ordenação, ferramentas de coordenação podem ser utilizadas em nível de aplicação para esse fim. Este artigo propõe uma integração do serviço de coordenação na arquitetura do sistema Kubernetes, um orquestrador de containers, com o objetivo de manter os containers os menores possíveis e oferecer à aplicação a capacidade de replicar estados automaticamente. Um protocolo que utiliza memória compartilhada disponível no Kubernetes foi desenvolvido para atuar na arquitetura, e uma avaliação foi realizada para demonstrar a viabilidade da proposta.*

Abstract. *Computer virtualization brought fast resource provisioning to data centers and also the deployment of a pay-per-use cost model. The system virtualization provided by containers has improved this versatility of resource provisioning. Applications that require more restrictive agreement and ordering warranties can use specific tools to achieve their goals. This paper proposes the integration of coordination services in the Kubernetes architecture in order to maintain containers as small as possible and offer automatic state replication. A protocol that uses shared memory available in Kubernetes was developed, and an evaluation was conducted to show the viability of the proposal.*

1. Introdução

A utilização de máquinas virtuais em *data centers* tornou prático o provisionamento dinâmico de recursos. Por consequência, foi possível implantar o modelo de custos sob demanda (*pay-per-use*). O número de usuários conectados à Internet cresce cada vez mais, fazendo surgir clientes em potencial para a variedade de serviços de computação disponíveis nos *data centers*. Esse conjunto de serviços disponíveis em *data centers* é conhecido como *cloud computing*, definido pelo Instituto Americano Nacional de Padrões e Tecnologia [Mell and Grance 2011] como "um modelo para permitir acesso ubíquo, conveniente e sob-demanda para recursos de computação compartilhados e configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com esforço de gerenciamento mínimo ou por um serviço de interação

com o provedor". No contexto de provisionamento, os *containers* popularizaram-se em 2013, tornando ainda mais rápida a capacidade de alocar e liberar recursos em *data centers*. A implementação de *containers* mais conhecida na atualidade é o Docker (`docker.com`).

O uso de *containers* pode causar um impacto positivo nos *data centers*. Para organizar esse desenvolvimento, várias empresas se reuniram, e em Julho de 2015 fundaram a *Cloud Native Computing Foundation* (`cncf.io`), cujos objetivos incluem estabelecer padrões para a operacionalização de *containers* em diversos provedores de nuvens. A Google utiliza *containers* há alguns anos e lançou, como um primeiro resultado da CNCF, o sistema Kubernetes para orquestrar *containers* Docker em *clusters*. O Kubernetes é *open-source*, tem sido desenvolvido colaborativamente e traz consigo conhecimento dos engenheiros do Borg [Verma et al. 2015], o atual gerenciador de *containers* da Google.

O Kubernetes possui a capacidade de replicar *containers* a fim de aumentar a disponibilidade da aplicação hospedada no *container*. Quando um *container* falha, o Kubernetes elimina o *container* faltoso e instancia outro a partir de uma imagem. Nesse processo, o estado da aplicação hospedada no *container* é perdido. Aplicações podem utilizar volumes externos para persistir o estado da aplicação, mas é preciso proteger estes volumes contra falhas. Além disso, no caso de replicar estados da aplicação, o acesso ao volume deve ser organizado pela aplicação para lidar com questões de concorrência.

Algumas aplicações requerem garantias mais restritivas de acordo e ordenação, como aplicações de trabalho cooperativo ou aplicações que realizam processamento replicado por razões de tolerância a faltas. Em ambientes de rede local, os algoritmos mais eficientes são derivados do Paxos [Lamport 1998]. O sistema ZooKeeper [Hunt et al. 2010] e o algoritmo Raft [Ongaro and Ousterhout 2014] são utilizados na prática para criar máquinas de estado replicadas [Schneider 1990]. Estas soluções podem ser utilizadas em nível de aplicação no Kubernetes, porém ocuparão espaço em todos os *containers* participantes da replicação, além de onerar a aplicação com o esforço de realizar a coordenação.

Para fornecer ao Kubernetes a possibilidade de lidar com estados replicados, a abordagem de integração [Lung et al. 2000] é apropriada pois está alinhada com a natureza colaborativa de desenvolvimento do Kubernetes. Integrar significa construir ou modificar um sistema existente, adicionando componentes para complementar funcionalidade. Outras abordagens são possíveis, como interceptação ou serviço, porém a integração é transparente ao usuário, sendo também a abordagem que apresenta melhor desempenho, segundo experiências realizadas no âmbito do CORBA [Lung et al. 2000]. Assim sendo, este trabalho apresenta duas contribuições. A primeira é uma arquitetura de suporte a replicação de máquinas de estado no sistema Kubernetes, utilizando um componente pré-existente no Kubernetes como memória compartilhada para orientar a coordenação entre *containers*. A segunda contribuição é um protocolo denominado ONSET (OrdeNação Sobre mEmória comparTilhada) que opera na arquitetura proposta.

O artigo está organizado da seguinte forma: a Seção 2 apresenta conceitos sobre *containers* e o sistema Kubernetes. Na Seção 3 é detalhada a proposta, em termos de protocolo e arquitetura. A Seção 4 apresenta a avaliação da proposta. Na Seção 5 são abordados os principais trabalhos relacionados e por fim a Seção 6 conclui o artigo.

2. *Containers* e Kubernetes

Certos componentes do Linux tornam possível realizar virtualização em nível de sistema. O *control group* (*cgroups*) permite controlar o acesso de recursos pelos processos. Um componente chamado *namespaces* permite isolar processos e recursos hierarquicamente em subníveis. *Ambiente* é o conjunto de processos e recursos, e ambientes criados em níveis inferiores não pode acessar ambientes de níveis superiores[1]. Por fim, um componente que consiste de um sistema de arquivos em camadas permite que dentro do ambiente isolado seja instalado um sistema operacional (SO), mas que efetivamente sejam gravados apenas os arquivos que não existem no SO do *host*. Este ambiente isolado, gerenciável e com arquivos em camadas é atualmente denominado *container*. O *cgroups* está no *kernel* do Linux desde 2007, mas foi reprojetado em 2013, fato que potencializou a criação de gerenciadores de *containers*, como o Docker. *Containers* são máquinas virtuais cujo estado não é persistente, e são instanciados a partir de imagens. As imagens são pequenas, pois utilizam sistema de arquivos em camadas. Por isto, a criação de *containers* é muito rápida e o provisionamento de recursos torna-se mais eficiente em comparação às máquinas virtuais tradicionais.

Uma rede que interliga *containers* hospedados em diferentes máquinas e o monitoramento de *containers* são funcionalidades necessárias em um *cluster*. A Google criou o sistema Kubernetes (`kubernetes.io`) a fim de integrar soluções para permitir o gerenciamento completo. Engenheiros da Google compõem a equipe principal do Kubernetes, que herdou conceitos do Borg [Verma et al. 2015], o atual gerenciador de *containers* da Google. No Borg há um componente chamado *Alloc* que mantém na mesma máquina *containers* que tem alto acoplamento e por isso precisam compartilhar recursos. Considere o exemplo de um servidor web e a aplicação que monitora/analisa os *logs* produzidos pelo servidor web. Estas aplicações podem estar localizadas em *containers* diferentes, de forma a facilitar a substituição ou a atualização de uma das aplicações. Ambos *containers* compartilham os *logs*, que são produzidos pelo servidor web e consumidos pelo analisador de logs. No Kubernetes, o componente que mantém *containers* relacionados próximos é o POD, e possui objetivos semelhantes ao *Alloc*. A Figura 1 ilustra a arquitetura do Kubernetes.

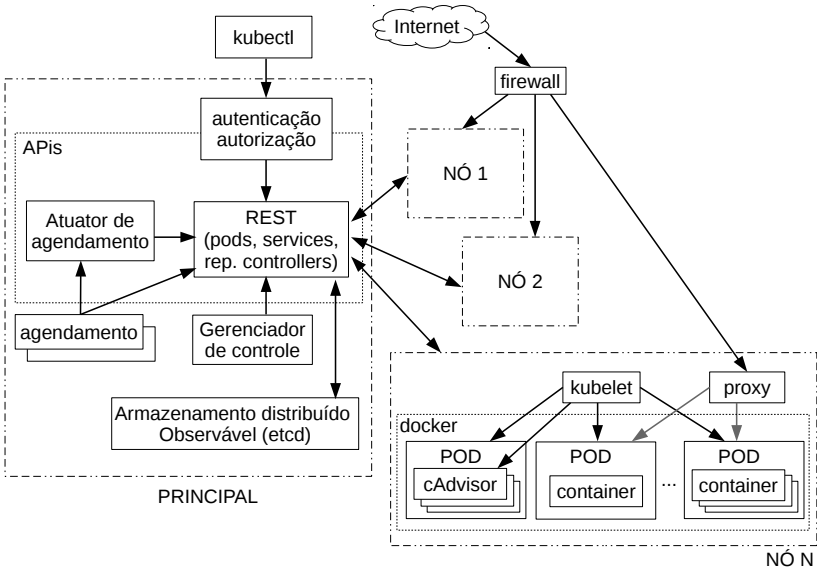


Figura 1. Arquitetura do Kubernetes.

[1]O comando `chroot` do Linux é um exemplo de implementação de espaço de nomes.

Um *cluster* Kubernetes é formado de nós, que são as máquinas (físicas ou virtuais) nas quais *containers* são hospedados. A máquina principal contém componentes gerenciais, e pode ser replicada para tolerar faltas de parada. O POD é uma unidade mínima de gerenciamento no Kubernetes, e pode conter um ou mais *containers*. PODs recebem endereço de rede e são alocados em nós. *Containers* que estão no mesmo POD compartilham recursos e são mantidos no mesmo nó. Clientes contactam o *cluster* por meio de um *firewall*, que geralmente distribui pedidos aos nós segundo uma estratégia de balanceamento de carga. O *proxy* recebe pedidos de clientes e encaminha os pedidos ao POD. Se o POD estiver replicado, um balanceamento de carga interno ao nó distribui aleatoriamente o pedido a uma das réplicas. O *kubelet* gerencia PODs, imagens, *containers* e outros elementos do nó. O *cAdvisor* monitora recursos utilizados pelo *container*. O *kubelet* repassa as informações de monitoramento à máquina principal, que deve atuar quando necessário.

As informações de controle do Kubernetes são armazenadas na máquina principal. O armazenamento de dados é observável, o que torna possível notificar clientes quando dados são modificados. A ferramenta *etcd* (`coreos.com/etcd`) implementa esse armazenamento no Kubernetes. Componentes acessam dados armazenados por meio de APIs REST. O *replication controller* mantém réplicas de PODs operantes, a despeito de falhas. O *kubectl* é uma *interface* de comando pela qual o operador pode criar PODs, verificar o estado do *cluster*, entre outras operações. O componente de agendamento determina em quais nós os PODs serão instanciados. O atuador de agendamento realiza a instanciação e a destruição de PODs. O gerenciador de controle atua em nível de *cluster*, realizando funções como descoberta e monitoramento de nós. Por fim, componentes de segurança permitem definir formas de autenticação e autorização de acesso do *kubectl* ao *cluster*.

A replicação de *containers* no Kubernetes utiliza monitoramento para eliminar ou criar *containers* e manter ativa a quantidade de réplicas definida. O balanceamento de carga realizado pelo *proxy* também favorece a disponibilidade da aplicação. Entretanto, existem aplicações que necessitam de garantias mais restritivas de acordo e ordenação. Aplicações de trabalho cooperativo ou aplicações que realizam processamento replicado por razões de tolerância a faltas são alguns exemplos. Estes requisitos podem ser alcançados com a utilização, em nível de aplicação (dentro dos *containers*), de ferramentas como o ZooKeeper [Hunt et al. 2010] e o algoritmo Raft [Ongaro and Ousterhout 2014]. Porém, retirar da aplicação a responsabilidade de realizar essa tarefa pode reduzir o tamanho da imagem do *container* e reduzir a complexidade no interior do *container*. O presente artigo é o primeiro passo nesta direção: integrar a coordenação de estados de *containers* ao Kubernetes.

O Kubernetes usa um armazenamento observável, implementado pela ferramenta *etcd*, para persistir as informações sobre o gerenciamento dos *containers*. É possível utilizar o *etcd* para coordenar pedidos, reaproveitando o recurso que já está disponível no Kubernetes. O *etcd* pode ser usado como uma memória compartilhada, e protocolos de replicação de estados apoiados em memória compartilhada geralmente apresentam baixo número de mensagens de rede, em comparação aos protocolos que não usam memória compartilhada. O *etcd* utiliza o RAFT para distribuir o armazenamento, tornando a memória compartilhada confiável e disponível. Um trabalho recente [Toffetti et al. 2015] utiliza o *etcd* de forma distribuída para persistir estado e agregar confiabilidade a microserviços.

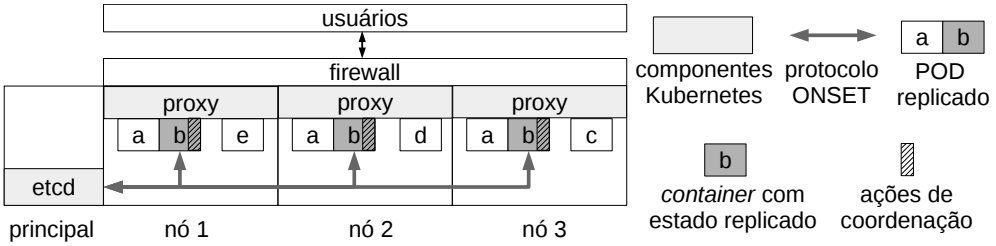


Figura 2. Arquitetura para Integração de Coordenação no Kubernetes.

3. Integração de Coordenação no Kubernetes

O Kubernetes é uma ferramenta *open-source*, o que possibilita o desenvolvimento de extensões de forma irrestrita. Proposta semelhante de modificação em arquitetura foi realizada no contexto do CORBA [Bessani et al. 2005], quando várias empresas reuniram-se para criar o padrão FT-CORBA. A proposta de incorporar coordenação de serviços na arquitetura Kubernetes considera a abordagem de integração [Lung et al. 2000], onde componentes podem ser modificados ou novos componentes podem ser adicionados à arquitetura.

A Figura 2 apresenta a proposta de integração de coordenação no Kubernetes. Nós são máquinas (físicas ou virtuais) do *cluster* Kubernetes. A máquina principal contém componentes de gerência do Kubernetes. Um *firewall* encaminha pedidos aos nós, e o *proxy* encaminha o pedido um *container* hospedado no nó. O *etcd* é o componente de armazenamento do Kubernetes, e será usado pelos *containers* como memória compartilhada. Nesta proposta, ações de coordenação encontram-se nos *containers* que possuem estado replicado.

3.1. Modelo de Sistema

O sistema é composto por *containers* replicados, identificados na Figura 2 pela letra `b`. Estes *containers* possuem estados replicados, devido à execução, na mesma sequência, de todos os pedidos que os *containers* recebem [Schneider 1990]. Assume-se que até f *containers* possam falhar por parada. Dessa forma, devem existir pelo menos $2f+1$ réplicas no sistema. A parada de nós também é tolerada, sendo que neste caso *containers* são realocados para outro nó automaticamente pelo Kubernetes.

A memória compartilhada (MC) é assumida como um repositório chave-valor que pode ser acessado por operações de leitura e escrita. *Containers* de estado replicado utilizam a MC como forma de comunicação. A MC possui capacidade de notificar clientes inscritos sobre eventos, como atualização de um valor e criação de novos valores. A MC também possui a capacidade de associar um identificador único sequencial a cada informação gravada. A MC pode ser distribuída em vários nós, no intuito de prover confiabilidade. A comunicação entre clientes e *containers* segue o modelo de sincronia parcial [Dwork et al. 1988], no qual existem períodos estáveis de comunicação síncrona. A comunicação entre os *containers* e a MC é assíncrona.

3.2. Interação entre componentes da arquitetura

O diagrama de sequência na Figura 3 mostra a interação do cliente com os *containers*. Considere b_i o *container* b replicado no nó i. No início, o cliente envia o pedido P, que será entregue a um dos nós pelo *firewall*. Dentro do nó, P é recebido pelo *proxy* e encaminhado a uma réplica do *container* b. No diagrama, b_1 recebeu P, o que significa que o *firewall* encaminhou P ao nó 1. A seguir, b_1 registra a chegada de P na memória compartilhada (MC), implementada pelo *etcd*. A MC notifica os demais *containers* replicados, que por

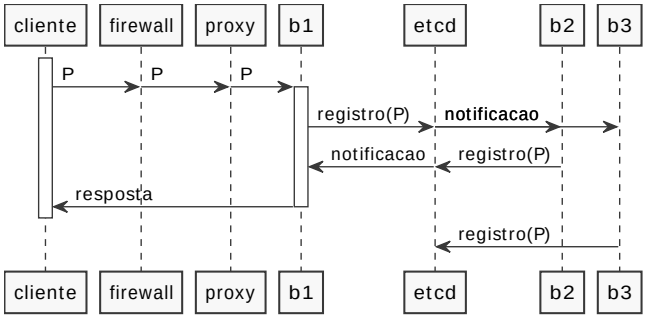


Figura 3. Interação na replicação de estados.

sua vez também farão seus registros de recebimento de P. A MC notifica b_1 sobre os registros feitos pelos demais *containers*. Quando b_1 acumula certo número de notificações, envia a resposta do pedido ao cliente (note que o registro de P por b_3 não é requisito para que b_1 responda ao cliente). A replicação de estados é garantida pois os *containers* replicados executam P na mesma ordem, por meio do protocolo de ordenação detalhado a seguir.

3.3. Protocolo de Ordenação

A ordenação de pedidos ocorre com suporte de memória compartilhada (MC), o que sugeriu o nome do protocolo: ONSET (OrdeNação Sobre mEmória comparTilhada). O ONSET está ilustrado na Figura 4. Qualquer réplica pode receber o pedido do cliente. Esta característica é importante pois o Kubernetes realiza balanceamento de carga automaticamente (em nível de nó, via componente *proxy*), e assim o protocolo torna-se compatível com esse recurso, não limitando o pedido ao recebimento apenas pelo líder. Caso a réplica receptora seja a réplica líder, uma ordem é associada ao pedido e um registro de chegada de pedido (RCP) é criado e inserido na MC. As demais réplicas serão notificadas e registrarão na MC seus respectivos RCPs, o que por sua vez fará com que a MC emita novas notificações. Quando uma réplica acumular $f + 1$ RCPs é possível executar o pedido. Apenas a réplica que recebeu o pedido via *proxy* responde ao cliente o resultado da execução.

Se uma réplica não-líder receber o pedido, essa réplica deve inserir o RCP na MC informando ordem zero, o que significa que o pedido está aguardando manifestação do líder para que seja definida a ordem do pedido. Quando o líder é notificado sobre esse pedido, o líder atribui uma ordem ao pedido e insere na MC o RCP do pedido, desta vez com uma ordem definida. A partir de então, o protocolo comporta-se de maneira semelhante ao caso no qual o líder recebe o pedido. Se o cliente não receber resposta da réplica após um tempo específico, o cliente deve reenviar o pedido. O reenvio poderá ser atribuído à mesma réplica ou a uma outra réplica, dependendo da política de balanceamento de carga utilizada.

A Figura 4 ilustra execuções do ONSET em cenários com 3 réplicas ($S_{0..2}$). Na Figura 4(a), a réplica S_0 é líder, responsável por definir a ordem dos pedidos. Números de sequência (1..10) na Figura 4(a) orientam a descrição que segue. O cliente inicia enviando o pedido P à réplica S_0 (1). A réplica S_0, por ser líder, atribui uma ordem à P e armazena na MC (2) o RCP de P, ou seja, $RCP(S_0, P)$. A MC responde S_0 confirmando a gravação de $RCP(S0, P)$ (3) e notifica as demais réplicas. Quando S_1 recebe a notificação sobre $RCP(S_0, P)$ (4), S_1 cria o seu próprio $RCP(S_1, P)$ e envia à MC (5). A MC responde S_1 (6) e notifica as demais réplicas (7 e 8) sobre $RCP(S_1, P)$. Quando S_1 recebe a resposta da MC (6), S_1 possui $f + 1$ RCPs e então pode executar P (o retângulo azul representa execução). Quando S_0 recebe $RCP(S_0, P)$ (7), S_0 pode executar P e responder ao cliente. Quando S_2 recebe $RCP(S_1, P)$ (8), S_2 pode executar P. Na Figura 4(a), a notificação

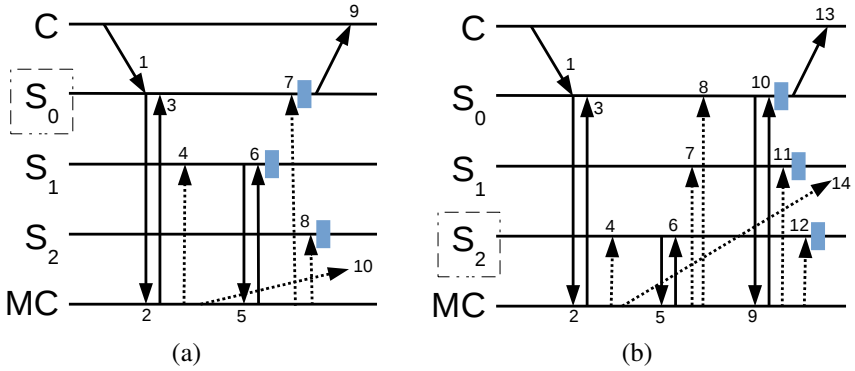


Figura 4. Protocolo ONSET: pedido enviado à uma réplica: (a) líder; (b) não-líder.

$RCP(S0, P)$ para S_2 foi intencionalmente atrasada (10) para mostrar que há casos onde S_2 não precisa enviar RCP à MC, e também para simplificar a ilustração do protocolo.

A Figura 4(b) mostra um cenário no qual um pedido P foi enviado à réplica não-líder S_0 (1). A réplica líder é S_2. Inicialmente, S_0 envia $RCP(S_0, P)$ à MC informando a ordem de P igual a zero. Pedidos com ordem zero permanecem na MC aguardando até que o líder atribua uma ordem. Quando a MC recebe $RCP(S_0, P)$ (2), responde à S_0 (3) e notifica as demais réplicas. Quando S_2 recebe $RCP(S_0, P)$ (4), atribui uma ordem a P e envia à MC $RCP(S_2, P)$ (5). Quando S_1 recebe $RCP(S_0, P)$, ações de comunicação não são executadas, pois S_1 não é líder. A MC responde S_2 (6) e notifica as demais réplicas. Ao receber $RCP(S_2, P)$ (8), S_0 envia $RCP(S_0, P)$ novamente à MC (9) (desta vez, com ordem definida pelo líder). Quando S_0 recebe a resposta da MC (10), S_0 possui $f+1$ $RCP's$, o que permite a S_0 executar P e responder ao cliente (13). Ao receber a notificação da MC, demais réplicas podem executar P (11 e 12). Na ilustração, a notificação de $RCP(S_0, P)$ foi intencionalmente atrasada, dispensando o envio de $RCP(S_1, P)$ pela réplica S_1.

O protocolo ONSET tolera faltas de paradas de réplicas. As Figuras 4(a) e 4(b) apresentam cenários onde a participação das réplicas S_2 e S_1, respectivamente, não influenciam no progresso do protocolo. Faltas de parada nas réplicas apresentariam comportamento semelhante aos casos de atraso representados na Figura 4.

3.3.1. Eleição

Protocolos que utilizam sequenciadores (fixo ou móvel) geralmente utilizam temporizadores para monitorar o funcionamento do líder. No ONSET, temporizadores são acionados no momento em que a réplica percebe a existência de um pedido, seja pelo recebimento direto do cliente ou através de notificações da MC. Se um temporizador expira e o líder ainda não definiu uma ordem para o pedido, a réplica envia à MC uma solicitação de eleição. A MC notificará demais réplicas, que registrarão solicitações de eleição, gerando notificações. Após $f + 1$ solicitações de eleição, uma réplica assume o papel de líder.

O critério de eleição para o novo líder é o seguinte: a primeira réplica que efetuou registro de eleição tornar-se-á o novo líder. Dessa forma, a réplica que possui um pedido atrasado há mais tempo provavelmente será a primeira a solicitar a eleição, e poderá tão logo possível tornar-se líder, ordenar o pedido e promover o progresso do protocolo. Uma observação relevante é que o primeiro líder só é eleito a partir do momento em que o sistema recebe o primeiro pedido. Para certificar-se de que todos os registros de eleição são

conhecidos da réplica (isso é importante para esquivar-se de possíveis inversões na rede de notificações da MC), ao acumular $f + 1$ solicitações de eleição a réplica lê da MC o conjunto completo de solicitações de eleição. Assim, é possível observar com exatidão (de maneira determinística) qual foi o primeiro registro de eleição. Esse critério torna-se possível graças ao identificador único sequencial gerado pela MC (Seção 3.1), cujo menor valor remete ao primeiro registro de eleição efetuado.

Outra característica importante do mecanismo de eleição adotado é que as réplicas não conhecem diretamente a réplica líder (há um certo anonimato), mas sabem de sua existência pela observação do progresso da ordenação dos pedidos. Essa característica é coerente com o ambiente de *containers*, no qual a criação e destruição de réplicas é mais frequente do que nos ambientes tradicionais de máquinas físicas ou virtuais. Apesar de não ser necessário para o funcionamento do protocolo, réplicas podem inferir quem exerce a liderança observando o autor do primeiro RCP de um pedido ordenado na MC.

4. Avaliação

A avaliação desta proposta foi realizada com a execução do protocolo ONSET sobre a arquitetura apresentada (Figura 2). Testes preliminares com o ONSET revelaram intensa atividade no disco do computador principal. Estas atividades em disco são oriundas do armazenamento realizado pelo *etcd*. Supôs-se então que a hospedagem do *etcd* em um diretório montado em memória RAM poderia favorecer o experimento, pois memória RAM é um dispositivo que possui latência e variabilidade de tempo de acesso menores do que disco rígido. Para investigar essa possibilidade, apresenta-se a primeira hipótese:

Hipótese 1: *memória compartilhada (MC) armazenando dados em disco virtual alocado em memória RAM trará resultados mais estáveis que execução da MC usando disco rígido. Ainda, o desempenho será pelo menos 50% melhor, comparado ao uso de disco rígido.*

Alocação de recursos no contexto de *containers* é menos custosa do que alocação usando máquinas virtuais tradicionais. Além disso, o protocolo ONSET possui poucas mensagens, em comparação a outros protocolos, por utilizar memória compartilhada como meio de comunicação entre réplicas. Dessa maneira, é razoável supor que a vazão geral do sistema (quantidade de pedidos executados por segundo) não será bruscamente afetada quando o número de réplicas for incrementado. Pode-se então enunciar a seguinte hipótese:

Hipótese 2: *usando containers, é possível utilizar um número de réplicas pelo menos 4 vezes superior ao cenário clássico de 3 réplicas, para tolerar faltas de parada. Haverá uma diminuição na vazão do sistema, mas esta será gradual, na medida em que o número de réplicas cresce, e não maior que 20% para cada incremento de f, na fórmula $n = 2f+1$.*

4.1. Experimentos

Experimentos foram conduzidos para avaliar as hipóteses apresentadas. Para avaliar a Hipótese 1, o computador principal do *cluster* Kubernetes foi dotado de um disco virtual de 4GB, montado em memória RAM, com uso do comando `mount` do Linux, e o sistema de arquivos `tmpfs`. Foi realizada uma execução do experimento com o *etcd* utilizando o disco rígido, e outra execução com o *etcd* utilizando o disco virtual. A fim de avaliar a Hipótese 2, o número de réplicas foi variado de 1 a 15, seguindo a regra de tolerância a faltas de parada: $n = 2f + 1$, onde n é o número de réplicas presentes no sistema e f é o número máximo de faltas toleradas. Adicionalmente, o número de clientes que realizam pedidos foi variado

de 1 a 64, dobrando-se o número de clientes a cada variação. Foram medidas a latência dos pedidos, em milissegundos, e a duração de cada cenário de execução do experimento, em segundos. Para quantidades de clientes superiores a 2, múltiplas *threads* simularam o acesso simultâneo de vários clientes. Os pedidos consistiram em requisições no estilo REST, com tamanhos que não excederam 50 bytes, e as respostas dos pedidos não excederam 100 bytes. Cada cliente efetuou 50 requisições ao sistema, considerando que um cliente apenas envia um próximo pedido após receber a resposta do pedido atual.

O protótipo do ONSET foi desenvolvido na linguagem de programação Go (`golang.org`), versão 1.4.2, visando uma futura integração no Kubernetes, que também é desenvolvido em Go. Um *cluster* Kubernetes foi montado para avaliar a execução do ONSET em *containers*. A aplicação mantém um contador compartilhado sobre o qual podem operar 3 comandos: *get* para retornar o contador; *inc* para incrementar o contador; e *dou*, uma operação que divide o contador por 2, se o contador for maior que 30. O temporizador que monitora o líder foi configurado para 1 segundo[2]. A aplicação está hospedada em uma imagem de *container* disponível no repositório `hub.docker.com`, sob o localizador `replicatedcalc`. O programas utilizados nos *containers* (réplicas) e no experimento (clientes) estão disponíveis no endereço `hylson.com/kubernetesSBRC2016`.

Os experimentos foram realizados em um *cluster* com quatro computadores com o sistema Kubernetes instalado, e um quinto computador atuando como cliente. Todos os computadores possuem configuração Intel i7 3.5Ghz, QuadCore, cache L3 8MB, 12GB RAM, 1TB HD 7200 RPM. Uma rede ethernet 10/100MBits isolada de tráfego externo conectou os computadores durante a realização dos experimentos. *Containers* que executam em máquinas distintas comunicam-se por meio de uma rede virtual, implementada neste trabalho com o Flannel (`github.com/coreos/flannel`). Os computadores possuem o sistema operacional Ubuntu Server 14.04.3 64 bits, com *kernel* versão 3.19.0-42.

4.2. Resultados e Discussões

A Figura 5 apresenta as latências médias percebidas pelos clientes, em função do número de réplicas executadas em *containers* no *cluster* Kubernetes. Na Figura 5(a) o experimento foi realizado utilizando memória compartilhada sobre um disco virtual em RAM, enquanto a Figura 5(b) refere-se à execução da memória compartilhada usando o disco rígido. Há uma diferença perceptível de desempenho quando o número de réplicas e clientes é pequeno. Especificamente, para poucos clientes (até 16) a utilização de memória RAM proporciona um desempenho melhor em relação ao cenário que utiliza disco rígido. Entretanto, a partir de 32 clientes, a diferença de latência percebida pelos clientes diminui, considerando as duas estratégias (disco virtual e rígido). Foram feitos testes estatísticos de hipótese sobre algumas medidas para aferir a diferença. Por exemplo, para as medidas de latências com 1 réplica e 64 clientes, a diferença entre o método que usa disco virtual em RAM e disco rígido é estatisticamente significativa, apesar de não ser aparente nos gráficos da Figura 5.

Além dos testes estatísticos, foram construídos gráficos específicos de alguns cenários, para melhor visualização. A Figura 6 mostra as latências percebidas por 64 clientes ao acessar 1 réplica (as latências de eleição de líder não estão representadas) e por 32 clientes ao acessar 15 réplicas. Os gráficos do tipo *boxplot* representam a mediana na linha central do retângulo, por isso foram também impressas as médias, representados pela linha de menor

[2]Este valor pode ser variado a fim de verificar limites.

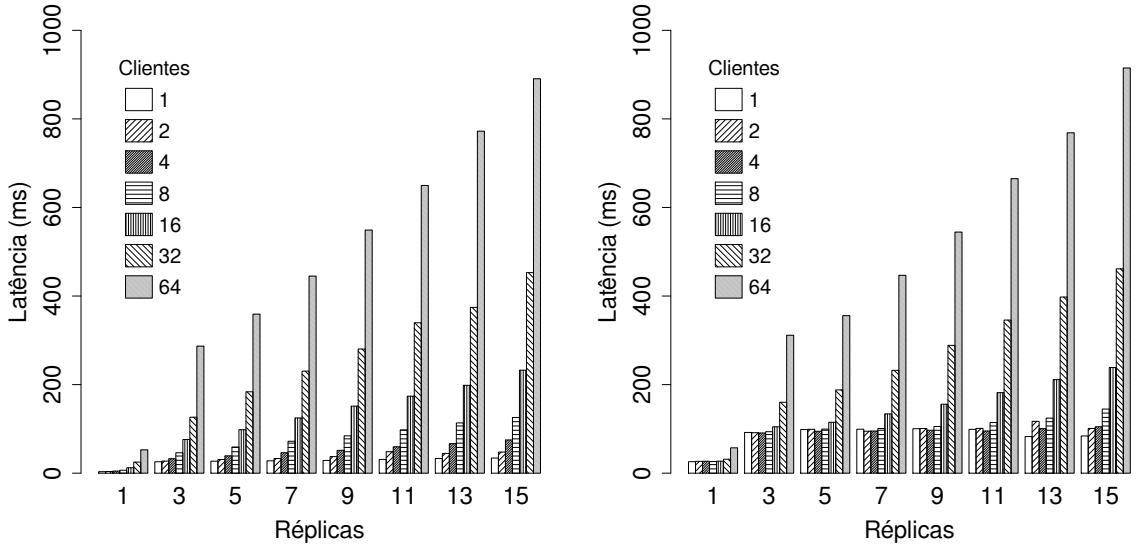


Figura 5. Latência com memória compartilhada em (a) disco virtual RAM, (b) disco rígido.

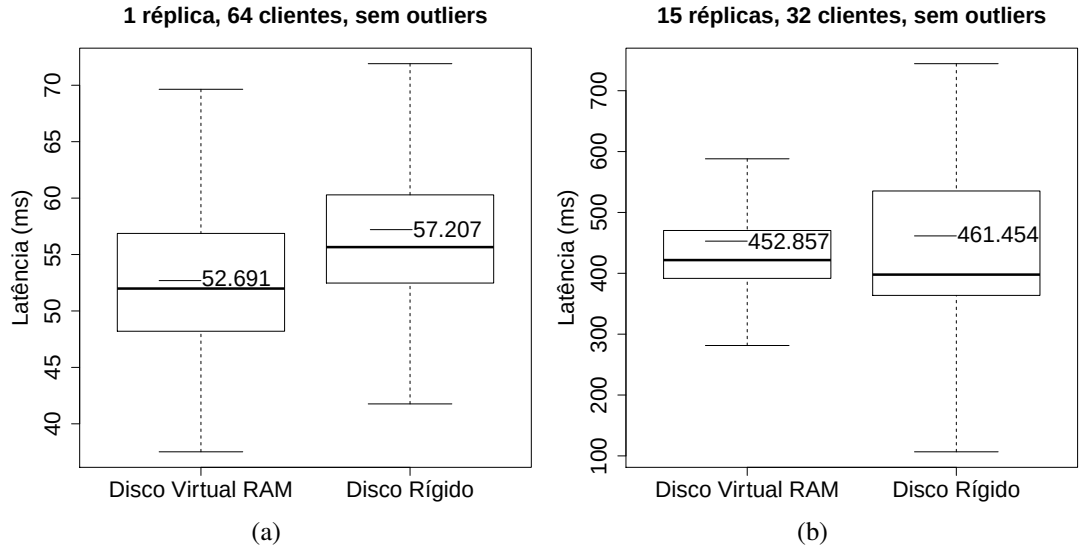


Figura 6. Latências para (a) 1 réplica e 64 clientes, e (b) 15 réplicas e 32 clientes.

comprimento, ao lado do valor da média. Em ambos cenários, a latência média com uso de disco virtual mostra-se menor do que a latência média com uso de disco rígido. Os testes estatísticos comprovam que no caso da Figura 6(a) é possível afirmar um melhor desempenho médio com uso de disco virtual em RAM sobre o disco rígido. Porém, no caso apresentado na Figura 6(b), os testes estatísticos não permitem afirmar a superioridade no uso do disco virtual sobre o disco rígido. Assim sendo, considera-se a Hipótese 1 como falsa por não haver evidências suficientes que comprovem o ganho de desempenho em todos os casos considerados. Os valores de média e desvio-padrão, bem como os testes estatísticos aplicados no exemplo, encontram-se disponíveis em `hylson.com/kubernetesSBRC2016`.

O líder do sistema apenas é eleito após a chegada do primeiro pedido. As latências percebidas pelos clientes destes primeiros pedidos podem ser consideradas medidas *outliers*, pois apresentarão valores superiores à média por conter em si o tempo de eleição do líder. Como exemplo, a Figura 7 mostra, para a execução de 3 réplicas usando memória compartilhada em disco rígido, pontos isolados de grande latência (*outliers*). Esses pon-

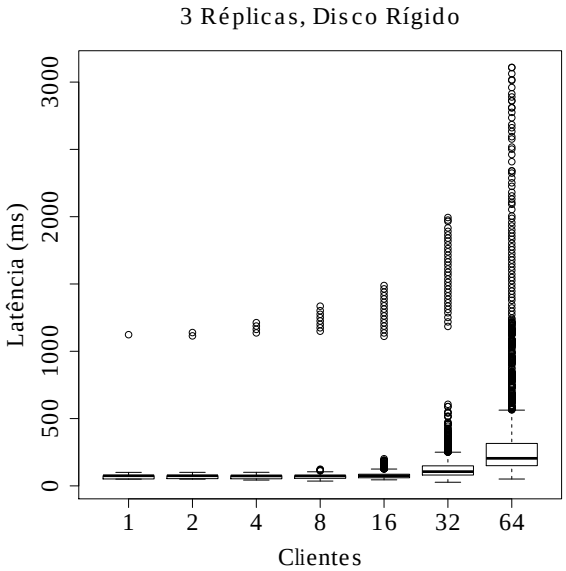


Figura 7. Latências: destaque da eleição de líder (*outliers*).

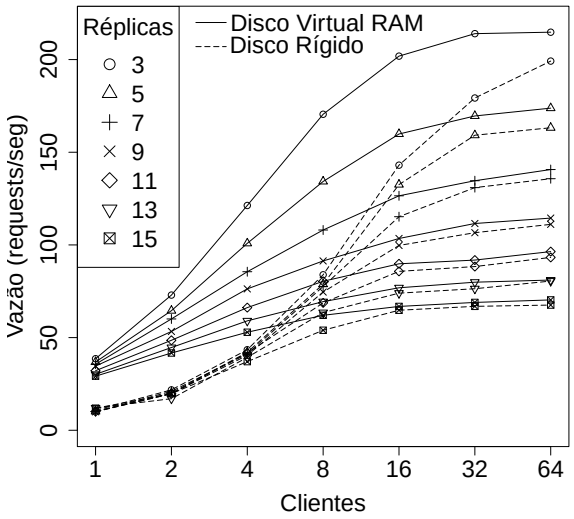


Figura 8. Vazão do sistema.

	Incremento de réplica (quantidades anterior/novo)						
	1/3	3/5	5/7	7/9	9/11	11/13	13/15
Mídia da MC	Percentual de redução da vazão						
Disco Virtual	84,7	16,42	15,05	13,48	12,58	11,7	10,13
Disco Rígido	74,93	8,51	6,8	8,98	7,27	6,86	6,81

Tabela 1. Percentuais médios de reduções de vazão ao incrementar número de réplicas.

tos em destaque carregam consigo, além do tempo de ordenação e execução do pedido, a duração da eleição do líder, que ocorre na chegada dos primeiros pedidos. No detalhe da Figura 7, é possível notar que para 4 clientes há 4 pontos em destaque, correspondente aos pedidos destes 4 clientes que perceberam maior latência devido à eleição do líder.

A Figura 8 apresenta a vazão do sistema em função do número de clientes, réplicas e forma de uso da memória compartilhada. Quando o número de réplicas cresce ocorre uma redução na vazão, mas esta perda diminui na medida em que o número de réplicas cresce, até a quantidade de réplicas avaliada no experimento (15). A Tabela 1 mostra a variação média (considerando a média de todos os números de clientes medidos no experimento) do percentual de vazão entre réplicas. Há uma grande redução na vazão quando a replicação é introduzida (mudança de 1 para 3 réplicas). Porém, nas mudanças subsequentes, a variação não é maior de 20%, conforme suposto na Hipótese 2. Portanto, pode-se considerar a Hipótese 2 verdadeira, pois foi possível experimentar um alto número de réplicas no sistema, e houve uma gradual redução na vazão durante o aumento do número de réplicas.

5. Trabalhos Relacionados

Tolerar faltas de parada em ambientes de rede local é um tema que possui bastantes estudos disponíveis na literatura. Entretanto, a aplicação de tolerância a faltas no contexto de *containers* ainda é incipiente. A seguir serão descritos alguns trabalhos recentes [Bolosky et al. 2011, Marandi et al. 2012, Moraru et al. 2013, Silva et al. 2013, Ongaro and Ousterhout 2014, Cui et al. 2015] e suas relações com a presente proposta.

Gaios [Bolosky et al. 2011] apresenta uma implementação de Paxos otimizada de forma alcançar um desempenho próximo do hardware. São toleradas faltas de parada e um

conjunto limitado de faltas Bizantinas. Gaios não persiste operações antes de executá-las e retornar o resultado ao cliente, com o objetivo de aumentar o desempenho. Porém, isso limita o número de faltas simultâneas que podem ocorrer pra que o estado não seja perdido ou corrompido. Além disso, a semântica resultante dessa otimização não é igual àquela usada pelos tradicionais bancos de dados. A presente proposta realiza a persistência das operações na memória compartilhada antes de executá-las e enviar a resposta ao cliente. É possível que uma operação proposta no Gaios nunca alcance quórum e não seja executada. No ONSET, todo pedido deve ser registrado na memória compartilhada por uma maioria de réplicas antes que a resposta seja enviada ao cliente. Desde que uma réplica consiga "contaminar" [Défago et al. 2004] a memória compartilhada com o pedido, este será executado em algum momento. A contaminação nesse caso é útil pois acelera o retorno de resultados para o cliente, caso o cliente perca a conexão com o sistema e precise re-enviar o pedido.

Multi-Ring Paxos [Marandi et al. 2012] é um protocolo projetado para ser escalável, aumentando a vazão do sistema à medida em que recursos são adicionados. O consenso é particionado em grupos, e cada grupo executa uma instância do algoritmo Ring Paxos[Marandi et al. 2010]. *Learners* que participam de mais de um grupo reúnem os resultados de diferentes grupos de maneira determinística. Por exemplo, um *learner* que participa dos grupos $g1$ e $g2$ reunirá as respostas usando uma estratégia *round-robin*: $r_1 1$, $r_1 2$, $r_2 1$, $r_2 2$, onde $r_i j$ é a resposta de ordem i advinda do grupo j. Para garantir o progresso de *learners* que participam em mais de um grupo, cada grupo controla o tempo que fica sem enviar mensagens. Após um tempo máximo, o grupo envia uma mensagem $SKIP$, informando que não possui novas instâncias de consenso, permitindo o progresso do sistema. O coordenador de cada anel monitora a taxa na qual mensagens são geradas, a fim de sugerir ajustes no monitoramento. A falha de um nó (*acceptor* ou *coordinator*) requer a formação de um novo anel. O Multi-Ring é um protocolo de *multicast* atômico, enquanto o ONSET utiliza uma única difusão de pedidos para ordenar pedidos e manter o estado replicado.

O EPaxos [Moraru et al. 2013] é basedo no Paxos, e tem por objetivos otimizar latência em rede amplamente distribuídas (WAN) e balancear carga entre réplicas. O consenso é alcançado com apenas 1 *round* de comunicação, quando pedidos concorrentes não interferem entre si. Neste caso, o protocolo funciona de maneira semelhante aos quóruns tradicionais [Malkhi and Reiter 1998]. Uma interferência entre dois pedidos A e B existe se a execução serial de A e B não for equivalente à execução serial de B e A. A aplicação é responsável por definir quais comandos interferem entre si. Quando pedidos concorrentes possuem interferência entre si, 2 *rounds* de comunicação são necessários. Como exemplo, considere que os pedidos A e B são enviados por diferentes clientes em um primeiro *round*. Um destes comandos será primeiramente executado por uma maioria de réplicas. Quando o comando B chegar a uma réplica X integrante da maioria de réplicas que executou A, esta réplica X irá informar a necessidade de executar A antes de ser possível a execução de B. O segundo *round* informará que $A \rightarrow B$, ou seja, a necessidade de executar A antes de B. Ocorre balanceamento de carga pois qualquer réplica pode atuar como líder de um pedido. Assim que possível, o líder notifica o cliente sobre a confirmação do pedido, e notifica as demais réplicas assincronamente. A latência é otimizada pois apenas um quórum de réplicas é suficiente para deliberar consenso. O EPaxos foi projetado para otimizar latência em WAN, mas o contexto de *containers* ainda é restrito a redes de área local (LAN).

RegPaxos [Silva et al. 2013] utiliza virtualização e compartilhamento distribuído de dados para criar uma memória compartilhada confiável. Dessa maneira, o número de ré-

plicas é reduzido de de $3f + 1$ para $2f + 1$ sem a utilização de componentes confiáveis complexos ou custosos em nível de software ou hardware. A presente proposta não tolera faltas bizantinas nem utiliza virtualização para criar a memória compartilhada, mas o ONSET é um protocolo que possui formato semelhante ao RegPaxos, exceto que no ONSET apenas uma réplica responde para o cliente. Isto foi assim definido porque o ONSET tolera apenas faltas de parada, dispensando a necessidade de enviar múltiplas respostas ao cliente.

Raft [Ongaro and Ousterhout 2014] é um protocolo de consenso desenvolvido para ser de fácil entendimento e, consequentemente, ser uma melhor base para o desenvolvimento de sistemas distribuídos. Os diferenciais do Raft são: i) o fluxo de entradas no *log* (registro de novos pedidos) segue apenas do líder para as demais réplicas; ii) o uso de temporizadores aleatórios na eleição de líder e nos mecanismos de *heartbeat* torna mais simples a resolução de conflitos; iii) uma nova abordagem permite com que o sistema continue operando normalmente mesmo durante mudanças de configuração. Raft possui desempenho e resultados equivalentes ao Paxos, porém devido ao objetivo de ser um protocolo de fácil compreensão, tornou-se amplamente utilizado. O ONSET também opta pela simplicidade ao utilizar uma memória compartilhada pré-existente no ambiente de comunicação entre as réplicas. O Raft mantém o sistema operante durante mudanças de configuração e requer um protocolo de mudança de visão. O ONSET também requer um protocolo de mudança de visão e continua operando durante mudanças de configuração, pois a memória compartilhada persiste os registros de chegada nas réplicas, independente de as réplicas permanecerem no sistema ou não. O registro das chegadas dos pedidos na memória compartilhada contém um número que indica o total de réplicas participantes, cuja atualização é suficiente para mudar o critério de maioria e assim alterar a condição da deliberação do consenso.

CRANE [Cui et al. 2015] é um protocolo de replicação de máquinas de estado que lida com a questão de não-determinismo provocado pela execução de pedidos em servidores com múltiplas *threads*. CRANE questiona a complexidade de usar interfaces de aplicações como ZooKeeper [Hunt et al. 2010] para coordenar a replicação. Assim, propõe tornar transparente a replicação, por meio de interceptação dos pedidos realizados em *sockets*, e da ordenação dos pedidos usando uma instância do Paxos. O não-determinismo causado pela execução em múltiplas *threads* é parcialmente resolvido por uma solução anterior dos mesmos autores, denominada PARROT. Um algoritmo chamado *time bubbling* orquestra a ordenação e a execução *multithread*, inserindo tempos de espera (”bolhas”) e preenchendo instantes de execução que estariam ociosos na ausência de requisições. Como as *bolhas* possuem tamanho fixo, o sincronismo global é garantido. CRANE usa um método de interceptação [Lung et al. 2000] para deixar a replicação de estados transparente, enquanto ONSET utiliza integração. Ambas abordagens tornam transparente a utilização de coordenação para a aplicação, mas a abordagem de integração registra desempenho melhor que a interceptação, conforme experiências realizadas no âmbito do padrão CORBA [Lung et al. 2000].

6. Conclusões

Este artigo apresentou uma arquitetura de integração de serviços de coordenação na plataforma Kubernetes, um sistema orquestrador de *containers*. Para operacionalizar a coordenação, foi desenvolvido o protocolo ONSET, que realiza ordenação com suporte da memória compartilhada já disponível no Kubernetes. Um protótipo foi desenvolvido na mesma linguagem de desenvolvimento do Kubernetes, para facilitar uma futura transição das ações de coordenação para um novo componente a ser integrado na arquitetura oficial do produto.

Experimentos demonstraram a viabilidade da proposta, a possibilidade em escalar réplicas em quantidades geralmente não consideradas ao utilizar máquinas (físicas ou virtuais) como réplicas, além da característica de gradual degradação de vazão durante o aumento no número de réplicas do sistema. O uso de *containers* pode provocar grande impacto nos *data centers*, e parte das aplicações que migrarem para essa plataforma poderão se beneficiar de serviços de coordenação oferecidos por protocolos como o ONSET. Empresas como a Google já utilizam *containers* há anos, e esse conhecimento está sendo disponibilizado agora também sob forma de *software* de código aberto.

Agradecimentos

Miguel Correia é bolsista CAPES/Brasil (projeto LEAD CLOUDS). O trabalho foi parcialmente financiado pela FCT UID/CEC/50021/2013 e pelo CNPq 455303/2014-2. Este trabalho foi também parcialmente financiado com recursos da FAPESC/IFC, processo/projeto Nº 00001905/2015, no qual Aldelir Luiz e Hylson Netto são integrantes.

Referências

Bessani, A. N., Lung, L. C., and da Silva Fraga, J. (2005). Extending the umiop specification for reliable multicast in corba. In *7th Int. Symposium on Distributed Objects and Applications*, Agia Napa, Cyprus.

Bolosky, W. J., Bradshaw, D., Haagens, R. B., Kusters, N. P., and Li, P. (2011). Paxos replicated state machines as the basis of a high-performance data store. In *USENIX NSDI*.

Cui, H., Gu, R., Liu, C., Chen, T., and Yang, J. (2015). Paxos made transparent. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 105–120. ACM.

Défago, X., Schiper, A., and Urbán, P. (2004). Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys*, 36(4):372–421.

Dwork, C., Lynch, N., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323.

Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. (2010). Zookeeper: Wait-free coordination for internet-scale systems. In *USENIX Annual Technical Conference*, volume 8, page 9.

Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169.

Lung, L. C., Fraga, J. S., Farines, J.-M., and Oliveira, J. R. S. (2000). Experiências com comunicação de grupo nas especificações fault tolerant corba. In *Simpósio Bras. de Redes de Computadores*, Belo H. - MG.

Malkhi, D. and Reiter, M. (1998). Byzantine quorum systems. *Distributed Computing*, 11(4):203–213.

Marandi, P. J., Primi, M., and Pedone, F. (2012). Multi-ring paxos. In *Dependable Systems and Networks (DSN), 42nd Annual IEEE/IFIP International Conference on*, pages 1–12.

Marandi, P. J., Primi, M., Schiper, N., and Pedone, F. (2010). Ring paxos: A high-throughput atomic broadcast protocol. In *International Conf. on Dependable Systems and Networks (DSN)*, pages 527–536. IEEE.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States. SP 800-145.

Moraru, I., Andersen, D. G., and Kaminsky, M. (2013). There is more consensus in egalitarian parliaments. In *Proceedings of the Twenty-Fourth Symposium on Operating Systems Principles*, pages 358–372. ACM.

Ongaro, D. and Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *Proceedings of USENIX Annual Technical Conference*, pages 305–320.

Schneider, F. B. (1990). Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

Silva, M. R. X., Lung, L. C., Luiz, A. F., and Magnabosco, L. Q. (2013). Tolerância a faltas bizantinas usando registradores compartilhados distribuídos. In *Simpósio Brasileiro de Redes de Computadores*, Brasília,DF.

Toffetti, G., Brunner, S., Blöchlinger, M., Dudouet, F., and Edmonds, A. (2015). An architecture for self-managing microservices. In *Workshop on Automated Incident Management in Cloud*, pages 19–24. ACM.

Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. (2015). Large-scale cluster management at google with borg. In *European Conference on Computer Systems*, page 18. ACM.

Trilha Principal do SBRC 2016
Sessão Técnica 3
Segurança em Redes

Uma metodologia para identificação adaptativa e caracterização de phishing

Pedro Henrique B. Las-Casas[1], Osvaldo Fonseca[1], Elverton Fazzion[1]
Cristine Hoepers[2], Klaus Steding-Jessen[2], Marcelo H. P. Chaves[2],
Ítalo Cunha[1], Wagner Meira Jr.[1], Dorgival Guedes[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

[2]CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança
NIC.br - Núcleo de Informação e Coordenação do Ponto BR

{pedro.lascasas,osvaldo.morais,elverton}@dcc.ufmg.br

{cristine,klaus,mhp}@cert.br

{cunha, meira, dorgival}@dcc.ufmg.br

Abstract. *Phishing remains one of the most significant Internet security problems, causing financial damage to organizations and users. This type of attack combines social engineering and other sophisticated techniques with which the attacker attempts to deceive the victim to steal personal information. Usually, the fight against phishing is done with some of the same techniques used to combat spam, such as those that focus on sets of words and characteristic terms. However, there are few studies that specifically address phishing and there are doubts about the stability and dynamics of message characteristics over time and how they can be leveraged to improve the system's defenses. In this work, we present an adaptive method to identify phishing messages and apply it on a large data set to identify and characterize hundreds of phishing campaigns.*

Resumo. *Phishing continua sendo um dos mais significativos problemas de segurança da Internet, causando prejuízos financeiros a organizações e usuários. Esse tipo de ataque combina engenharia social e outras técnicas sofisticadas com as quais o atacante tenta ludibriar a vítima para roubar informações pessoais desta. Usualmente, o combate ao phishing é feito com algumas das mesmas técnicas usadas no combate ao spam, como aquelas que focam em conjuntos de palavras e termos característicos. Entretanto há poucos trabalhos que tratam especificamente de phishing e há dúvidas sobre a estabilidade ou evolução de características das mensagens ao longo do tempo e sobre como elas podem ser exploradas para melhorar as defesas do sistema. Neste trabalho apresentamos um método adaptativo para identificação de mensagens de phishing e aplicamos este método em um grande conjunto de dados para identificar e caracterizar centenas de campanhas de phishing.*

1. Introdução

Atualmente, phishing é uma das atividades criminosas mais lucrativas na Internet. Um estudo de 2007 mostra que ataques de phishing nos Estados Unidos causaram prejuízos

de mais de 3,2 bilhões de dólares, afetando cerca de 3,6 milhões de usuários. De acordo com um relatório da Kaspersky Lab, de junho de 2012 à junho de 2013, 37,3 milhões de pessoas reportaram terem sido vítimas de tais ataques, representando um aumento de 87% em relação ao ano anterior. Phishing combina engenharia social e técnicas de ataque sofisticadas para enganar usuários, levando-os a acreditar que as entidades em questão são legítimas, de forma a roubar dados pessoais destas vítimas. Em geral, atacantes tentam ludibriar os usuários, mascarando emails e páginas web com marcas confiáveis para roubar informações sensíveis das vítimas, como senhas, números de identificação pessoal e dados do cartão de crédito.

Apesar de haver um grande esforço por parte da comunidade mundial para combater esse tipo de fraude [Fette et al. 2007, Kumaraguru et al. 2007, Bergholz et al. 2010], o fato notório é que o *phishing* ainda persiste, o que nos leva a crer que o *phisher*[1] evolui suas técnicas para ludibriar os diversos métodos de mitigação criados. Essa motivação deixa visível que tão importante quanto construir ferramentas para combater o *phishing*, é tentar entendê-lo em sua essência para permitir a evolução e o aprimoramento de mecanismos que o combatam. Essa atividade, aparentemente simples, demanda uma grande complexidade para obtenção de um conjunto de dados representativos o suficiente para tornar o trabalho relevante para a comunidade, bem como para criação ou adaptação de técnicas de aprendizado de máquina.

Neste trabalho, buscamos entender as principais características de *phishing* que norteiam sua disseminação atual utilizando 13 *honeypots* de baixa interatividade distribuídos em diferentes pontos do mundo. Esses *honeypots*, configurados para parecerem servidores de emails vulneráveis, coletaram, durante um período de 91 dias, 1,13 bilhões de mensagens de emails mescladas entre duas categorias: *spam* e *phishing*. Para separar os dois grupos, utilizamos termos característicos de *phishing* como “banco”, “clique” e “crédito” e comparamos com os termos representativos de cada email, obtidos com TF-IDF (*Term Frequency–Inverse Document Frequency*). Para cada mensagem, assinalamos um escore de compatibilidade entre as palavras do email e de *phishing* e comparamos a um limiar mínimo global para classificar o email como pertencente ao conjunto de *phishing*.

A primeira contribuição do trabalho é mostrar uma lista atualizada de palavras relacionadas a *phishing* encontradas em nossa base de dados, de forma a elucidar os principais tipos de técnicas utilizadas pelos *phishers*. Estando de acordo com nosso argumento inicial, onde caracterizamos o *phisher* como um personagem dinâmico, utilizamos as técnicas encontradas em [Mikolov et al. 2013] para expandir nossa lista de palavras, obtida na literatura, com outros termos que estejam relacionados aos termos iniciais. Com essa simples expansão, pouco mais de um milhão de mensagens que continham apenas uma palavra do conjunto inicial e que teriam, portanto, um escore baixo, agora tem seus escores maiores devido à adição das novas palavras e serão classificadas como *phishing* com maior precisão. Acreditamos que essa aboragem seja essencial para melhor diferenciar as mensagens de *phishing* das demais.

Fomos ainda capazes de apresentar características interessantes do phishing, como referente ao seu envio, majoritariamente através do protocolo SMTP. Além disso, mostramos que poucos endereços IP e poucas campanhas são responsáveis pela maior parte dos

[1] Aquele que envia o *phishing*.

emails relacionados ao *phishing*, mostrando que, ao combater estes poucos casos, torna-se possível reduzir consideravelmente os prejuízos causados por estes ataques.

Este trabalho está estruturado da forma que se segue. Na Seção 2 apresentamos trabalhos relacionados aos ataques de *phishing*. Na Seção 3 descrevemos, a um nível mais refinado, os coletores de mensagens e o processo para separar as mensagens de *spam* e *phishing*. Na Seção 4, focamos nas principais características do *phishing* e as principais campanhas encontradas em nossa base de dados. Concluimos o trabalho na Seção 5, mostrando os trabalhos futuros que podem ser feitos utilizando os resultados deste trabalho como inspiração.

2. Trabalhos Relacionados

Phishers utilizam diversas técnicas para ludibriar suas potenciais vítimas. Alguns exemplos das técnicas utilizadas são *DNS cache poisoning*, sequestro de servidores web, imitação de serviços Web ou engenharia social; enganando o usuário e roubando suas informações. Neste trabalho, focamos no último caso, analisando mensagens que buscam aparentar legítimas. Uma forma comum de proteção contra ataques de phishing é o uso de *blacklists*. Entretanto, *blacklists* não provêem proteção durante o ínicio dos ataques de *phishing*, pois a URL ou site deve ser inserido na *blacklist* para que essa forma de proteção comece a funcionar. Além disso, como as URLs utilizadas pelos *phishers* possuem tempo de vida curto, a utilização de *blacklists* acaba sendo ineficiente. De acordo com [Sheng et al. 2009], 63% das campanhas de phishing analisadas pelos autores duraram menos de duas horas.

Com relação às técnicas baseadas no conteúdo para detecção de phishing, Marchal et al. analisa as características das URL's presentes na mensagem para classificá-la como phishing [Marchal et al. 2014]. Entretanto, o método proposto não é capaz de analisar URL's encurtadas, que se mostram cada vez mais presentes. Como analisamos todo o conteúdo da mensagem, nosso método é capaz de identificar mensagens que apresentam tais URL's. Zhang et al. propôs CANTINA [Zhang et al. 2007, Xiang et al. 2011], um método utiliza TF-IDF para extrair os principais termos de um documento, checa-os em máquinas de busca e então, caso o domínio do documento em questão apareça na busca, a página é considerada legítima. Em nosso trabalho, utilizamos o algoritmo TF-IDF para identificar os e-mails de phishing, como mostrado na Seção 3. Em [Aggarwal et al. 2014], os autores utilizam processamento de linguagem natural para identificar e-mails contendo *phishing*. Em sua abordagem, os autores identificam características chave deste tipo de mensagem, como a presença de referências a dinheiro, e utilizam estas características para classificar as mensagens como *phishing*. As características propostas pelos autores foram utilizadas em nosso método de detecção dos e-mails contendo *phishing*.

3. Metodologia

Nesta seção explicamos como realizamos a captura das mensagens de e-mail, o processo de filtragem e as técnicas utilizadas para identificar mensagens de *phishing* e separá-las dos demais spams.

3.1. Coleta das mensagens

As mensagens foram capturadas por treze honeypots de baixa interatividade [Steding-jessen et al. 2007] instalados em diferentes países, sendo dois no Brasil, dois

nos Estados Unidos, e um coletor em cada um dos seguintes países: Argentina, Áustria, Austrália, Chile, Equador, Hong Kong, Holanda, Noruega, Taiwan e Uruguai. Os honeypots são configurados para simular servidores vulneráveis, como *proxies* HTTP e SOCKS e *relays* SMTP abertos. Quando um spammer se conecta ao servidor SMTP de um honeypot, ele é levado a crer que está interagindo com um servidor SMTP operando como um *relay* aberto. Quando uma máquina se conecta a um honeypot através dos protocolos HTTP ou SOCKS, é levada a crer que é capaz de estabelecer conexões com outros servidores SMTP na rede. Com a finalidade de esconder a identidade dos *spammers* e evitar que suas máquinas sejam colocadas em *blacklists*, *proxies* e *relays* são frequentemente utilizados para o envio de spam. Além disso, como esses protocolos são orientados a conexão, é improvável a ocorrência de IP *spoofing*, que só seria possível se feito ao longo da rota de retorno dos pacotes e durante toda a duração da conexão.

Como os *honeypots* não prestam serviço para nenhuma rede e não são anunciados publicamente, assumimos que todas as mensagens recebidas por eles provém de *spammers*. Toda interação com os *honeypots* é registrada e as mensagens são armazenadas localmente, e, diariamente, copiadas para os servidores centrais do projeto. Nossos *honeypots* nunca encaminham as mensagens recebidas, com exceção daquelas mensagens cujos conteúdos indicam, de acordo com regras pré-definidas[2], que são mensagens de teste utilizadas pelos *spammers* para verificar se *proxies* e *relays* abertos estão funcionando. Neste trabalho analisamos mensagens coletadas entre 01/08/2015 e 31/10/2015.

3.2. Pré-processamento da base de spam

Como possuímos pontos de coleta em diferentes pontos do mundo, nossa base de spam é composta por mensagens dos mais variados idiomas. Porém, como neste trabalho objetivamos analisar o conteúdo das mensagens para identificar aquelas que caracterizam phishing, optamos por selecionar apenas as mensagens em inglês. Cabe ressaltar que nossa metodologia é extensível aos demais idiomas.

Para realizarmos a identificação do idioma das mensagens, extraímos o corpo das mensagens, removendo tags HTML e URLs, bem como desconsiderando o conteúdo dos anexos das mensagens. Uma vez que as mensagens estavam tratadas, pudemos identificar o idioma atráves de uma biblioteca[3] baseada no classificador *Naive Bayes*. A biblioteca utilizada retorna todas as linguagens possíveis para determinado documento. Selecionamos todos aqueles que possuiam inglês dentre os possíveis idiomas. De um total inicial de 1.133.874.435 mensagens, reduzimos o corpo da base de dados para 13.213.796 mensagens em inglês, uma vez que o tráfego de spam é predominantemente composto por mensagens asiáticas. Por fim, o último passo do pré-processamento foi remover *stopwords*, colocar as palavras em caixa baixa, desconsiderar pontuação e eliminar caracteres não UTF-8, de forma a melhorar o desempenho das técnicas de processamento de linguagem natural utilizadas para identificação das mensagens de phishing.

3.3. Identificação das mensagens de phishing

Nosso método para identificação de mensagens de phishing em um conjunto de spams é baseado em técnicas de processamento de linguagem natural. O primeiro passo da

[2]Por exemplo, verificando presença de texto específico no assunto ou corpo da mensagem.

[3]https://pypi.python.org/pypi/langdetect

técnica é a identificação de características chave presentes em mensagens classificadas como phishing. A partir das características propostas por [Aggarwal et al. 2014], identificamos outras capazes de diferenciar phishing das demais mensagens. Tais características foram divididas nas seis categorias descritas a seguir:

Tratamento: Termos utilizados pelos *spammers* para se aproximar e ganhar a confiança da vítima.

Menção monetária: Uma forma comum utilizada por atacantes para convencer usuários a responder seus e-mails é a promessa de dinheiro fácil. Uma vez que a vítima acredita que existe a possibilidade de ganhar dinheiro, ela pode responder ao e-mail enviando suas informações.

Senso de urgência: Os atacantes tentam induzir a vítima a responder a mensagem o mais rápido possível. Uma das razões para isso é que, quando uma pessoa está sob pressão, ou em uma situação de urgência, ela usualmente perde parte de sua razão lógica, tendendo a tomar decisões precipitadas. Além disso, outra razão está no fato de que, o quanto antes a mensagem for respondida, menor a probabilidade da mensagem ou suas URLs terem sido inseridas em alguma *blacklist*.

Pedido de resposta: Outra característica identificada nas mensagens é o pedido de resposta. Como o atacante objetiva obter informações sensíveis do usuário, é necessário que o usuário responda a mensagem (ou acesse alguma URL presente nesta). Portanto, o *phisher* tenta convencer a vítima a responder o e-mail utilizando alguns termos característicos como “reply”, “response”, “answer”, entre outras listadas na Tabela 1.

Formulário: Verificamos que um número grande de mensagens pedem aos usuários para preencher e enviar um formulário com as informações, meios muito pouco utilizados por mensagens de *spam* não-*phishing*.

Segurança: Outra forma de atrair o usuário é mencionar invasão e bloqueio de contas. Como são acontecimentos reais e comuns, esse tipo de abordagem é eficiente.

A seguir apresentamos a nossa metodologia de determinação de phishings e suas respectivas campanhas, que pode ser sintetizada nos 6 passos descritos a seguir:

1. Determinação do conjunto inicial de termos Utilizando os resultados da literatura, identificamos um conjunto inicial de termos que é mostrado na Tabela 1, que é representativo, mas incompleto tendo em vista a evolução das estratégias de phishing e novos contextos que podem ser utilizados, como por exemplo as olimpíadas. Para sermos capazes de evoluir conforme novas abordagens vão sendo usadas pelos *phishers*, utilizamos a técnica Word2Vec [Mikolov et al. 2013] para aumentar o conjunto de palavras utilizado na identificação de *phishing* e torná-lo mais dinâmico, como descrito a seguir.

2. Expansão do conjunto de termos Com base nos vários textos (no nosso caso mensagens), Word2vec identifica palavras que são mais semelhantes e que estão mais relacionadas aos termos de entrada, que são os termos do conjunto inicial. Assim, para cada um dos termos do conjunto inicial, avaliamos a sua saída do Word2Vec e selecionamos os termos associados ao envio de *phishing*. Considerando a base utilizada neste artigo, aumentamos o conjunto de palavras usado na identificação de *phishing* em 156,25%. A

Tabela 1 apresenta os termos adicionados para cada categoria.

Tabela 1. Termos associados a phishing

Categoria	Conjunto Inicial	Termos Adicionados
Tratamento	dear, friend, hello, please	congratulate, valuable, entrusted, congrats, sponsored, nontransferable, expires, regards, authentic, apologize, thank, inconvenience
Menção a dinheiro	bank, money, cash, dollar	credit, customer, funding, purchase, $, transfer, payment, millionaire, profits, accountability, dollars, donate
Pedido de resposta	write, contact, reply, response, foward, send	communication, reapproved, reconfirm, confirming
Urgência	now, today, instantly, straightaway, directly, urgently, urgent, desperately, immediately, soon, shortly, quickly	important
Formulário	form, attach, attached, attachment	information, address, occupation, documentations, subscriber, confidential, zipcode
Segurança	security, violated	detected, correct, authorised, unauthorized, sign, reauthenticate, reliance, spamfiltered, recover, impostors, reactivate, suspects, account, verification

3. Escore da mensagem por categoria O próximo passo da nossa metodologia é estimar a pertinência de cada mensagem a cada categoria de phishing. Essa estimativa é quantificada por um escore baseado no princípio TF-IDF [Baeza-Yates e Ribeiro-Neto 1999] A parte TF do escore deve refletir a ocorrência de termos das categorias, ou seja, quanto mais termos de uma categoria ocorrerem, maior a chance da mensagem pertencer a essa categoria. Para cada mensagem msg e categoria cat, definimos $numtermo_{msg,cat}$ como o número de ocorrências dos termos de cat em msg. Calculamos então $TF_{msg,cat}$ como a razão entre $numtermo_{msg,cat}$ e o maior $numtermo$ que ocorre em msg: $TF_{msg,cat} = numtermo_{msg,cat}/Max_{c \in categoria} numtermo_{msg,c}$. A parte IDF do escore reflete a popularidade das categorias, ou seja, ele é a razão entre o logaritmo do total de mensagens na base ($log(nummsg_*)$) e o número de mensagens assinaladas à categoria cat ($nummsg_{cat}$): $IDF_{cat} = log(nummsg_*)/nummsg_{cat}$. Finalmente, o TF-IDF de uma categoria cat para uma mensagem msg é obtido pela multiplicação das partes.

4. Escore da mensagem O passo seguinte é gerar um escore para cada mensagem que é a soma dos $TF-IDF$ das categorias e a constante α, que quantifica outros aspectos da mensagem que estão comumente associados a phishing. No nosso caso, utilizamos apenas a informação do destinatário para determinar α. Assim, se a mensagem tem apenas um destinatário, α é 1, senão 0, de acordo com a lógica do phishing ser uma mensagem pessoal. Outros critérios como os discutidos na caracterização poderiam ser incorporados ao α. Assim, a equação a seguir determina o escore de cada mensagem msg:

$$ESCORE_{msg} = \alpha + \sum_{cat \in categoria} TFIDF_{cat}$$

5. Classificação de phishing O escore de cada mensagem pode ser utilizado para classificá-la como phishing ou não. Um aspecto importante neste caso é qual o limiar lim que separa os dois grupos com maior acurácia. Para tal, vamos utilizar uma metodologia baseada na curva ROC e na medida AUC [Brown e Davis 2006], que nos permite determinar o valor do escore que maximiza a taxa de acerto numa classificação. Na prática, selecionamos 800 mensagens aleatoriamente da nossa base, considerando um nível de confiança

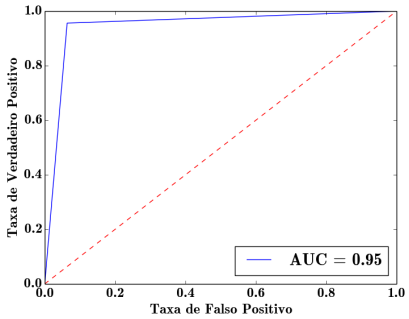


Figura 1. Curva ROC.

de 95% e um erro de $\pm 3,5\%$. A seguir rotulamos manualmente essas mensagens como phishing ou não-phishing. Construímos então a curva ROC (Figura 1) com as mesmas e seus escores e determinamos lim como sendo $1,5$ (ou seja, se $ESCORE_{msg} > lim$, então a mensagem é classificada como *phishing*), o que permite atingir uma AUC de 94,9%.

6. Determinação de Campanhas A partir do conjunto de mensagens classificadas como phishing, agrupamos as mesmas em campanhas (ou seja, grupos de mensagens com a mesma finalidade). O algoritmo utiliza o princípio $TF - IDF$, mas calculado no universo de todos os termos das mensagens. Assim, para cada mensagem msg e termo $termo$, definimos $numtermo_{msg,termo}$ como o número de ocorrências de $termo$ em msg. Calculamos então $TF_{msg,termo}$ como a razão entre $numtermo_{msg,termo}$ e o maior $numtermo$ que ocorre em msg: $TF_{msg,termo} = numtermo_{msg,termo}/Max_{t \in msg} numtermo_{msg,t}$. A parte IDF do escore é o logaritmo da razão entre o total de mensagens na base e o número de mensagens que contem $termo$: $IDF_{termo} = log(nummsg_{*}/nummsg_{termo})$. Finalmente, o $TF - IDF$ de $termo$ para uma mensagem msg é obtido pela multiplicação das partes. Esses valores de $TF - IDF$ compoem um vetor que é comparado posição a posição, com vistas a agrupar mensagens que possuem similaridade maior que 80%. Após desconsiderar as campanhas que possuem menos que 10 emails, encontramos 612 campanhas, que correspondem a mais de 8,5 milhões de mensagens. A Seção 4 apresenta uma caracterização dos resultados da nossa metodologia.

4. Resultados

Nesta seção apresentamos os resultados obtidos ao caracterizar os dados de *phishing* obtidos através da metodologia descrita anteriormente. Inicialmente, mostramos a visão geral dos dados de *phishing*. Em seguida, apresentamos as principais campanhas e as características apresentadas por cada uma delas.

4.1. Visão Geral

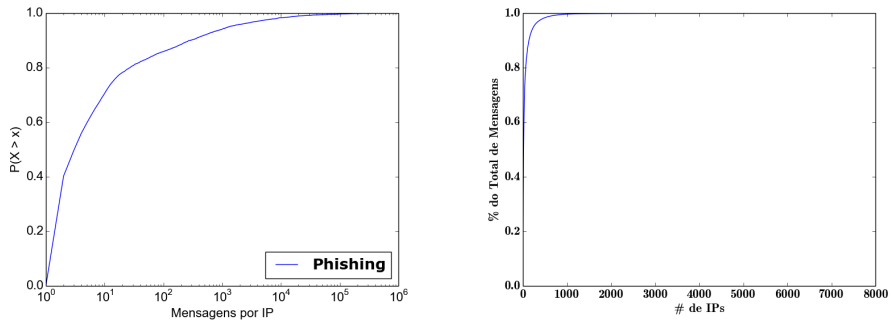
A Tabela 2 oferece uma visão geral das mensagens identificadas como *phishing*. No período de 91 dias, identificamos cerca de 9,7 milhões destas mensagens. Uma característica interessante destes e-mails está na forma como eles são enviados, basicamente através do protocolo SMTP, sendo mais de 99,9% do total das mensagens e mais de 99%

Tabela 2. Visão Geral de *Phishing*

	SMTP(%)	SOCKS(%)	HTTP(%)	Total
Mensagens	9.757.096 (99,94%)	4.550 (0,04%)	807 (0,0%)	9.762.453
Endereços IP	6.651 (99,22%)	52 (0,77%)	4 (0,0%)	6.703
Sistemas Autônomos (AS)	1.701(99,35%)	35 (2,04%)	4 (0,23%)	1.712
***Country Codes* (CC)**	154 (100%)	16 (10,38%)	3 (1,94%)	154

dos endereços IP responsáveis pelo envio das mensagens utilizando o SMTP. Os Sistemas Autônomos relacionados à origem das mensagens apresentam características semelhantes às dos endereços IP. Tal fato advoga em favor da gerencia da porta 25 afinal, desta forma, estas mensagens seriam bloqueadas evitando a propagação de milhões de mensagens de phishing.

O gráfico 2 mostra a distribuição das mensagens pelos endereços IP. Como é possível notar, a maior parte dos endereços enviaram poucas mensagens. Cerca de 60% enviou 10 mensagens ou menos e além disso, como pontuado anteriormente, utilizou-se o protocolo SMTP, característica de máquinas pertencentes à botnets, como mostrado em [Las-Casas et al. 2013]. Outro ponto interessante está na concentração de mensagens em alguns poucos endereços IP. O gráfico 2(b) mostra que 40% destes e-mails estão concentrados em apenas 10 endereços IP e que 90% dos phishing foram enviados de 100 endereços IP distintos. Portanto, uma forma de mitigar grande parte do tráfego de phishing e reduzir possíveis prejuízos causados por estas mensagens é combater estes principais atacantes, responsáveis pela maior parte dos ataques.



(a) CDF das mensagens por endereço IP (b) CDF da concentração das mensagens por IP

Figura 2. Análise do número de mensagens de phishing enviadas por endereços IP distintos.

Considerando estes poucos endereços IP responsáveis por grande parte das mensagens, apresentamos na Tabela 3 os 5 endereços IP que mais enviaram mensagens de phishing. Estes endereços IP possuem características semelhantes quanto ao baixo número de campanhas enviadas. O endereço IP 23.31.87.109, mapeado nos Estados Unidos, foi responsável pelo envio do maior número de mensagens em toda a base analisada. Este atacante enviou mais de 730 mil phishings durante os 3 meses considerados. Um ponto interessante é que, apesar do alto número de mensagens, não há grande variação no tipo de phishing enviado por este atacante, apresentando apenas 6 campanhas distintas. Estas campanhas, apesar de distintas, possuem o objetivo comum de ludibriar a vítima, oferecendo-a dinheiro para, desta forma, conseguir roubar seus dados pessoais.

Os endereços IP 212.227.94.138 e 212.227.255.64 encontram-se no mesmo AS, localizado na Alemanha. Estes endereços possuem uma campanha de phishing em comum (campanha 2, Seção 4.2), levando-nos a crer que são controlados por um mesmo *phisher*. Das mensagens enviadas por estes endereços, 74% e 71%, respectivamente, fazem parte desta campanha. Como será mostrado mais a frente, esta foi a segunda maior campanha encontrada em nossa base de dados e relaciona-se ao acesso a uma conta bancária.

Tabela 3. Top 5 Endereços IP

IP	# de Mensagens	AS	CC	# de Campanhas
23.31.87.109	732.361	7922	US	6
212.227.94.138	524.056	8560	DE	2
65.29.192.68	500.351	10796	US	1
212.227.255.64	384.054	8560	DE	2
186.83.40.72	368.498	10620	CO	2

Com relação à distribuição geográfica de origem, a maior parte das mensagens de *phishing* (mais de 60%) provêm dos Estados Unidos e da Alemanha. Além de ser o país com maior número de mensagens de *phishing*, os Estados Unidos também são aqueles que apresentam o maior número de campanhas distintas (233). Entretanto, estas campanhas possuem um média relativamente baixa de mensagens (15.476), principalmente quando comparado à Alemanha. Este país possui apenas 33 campanhas distintas, mas apresenta a elevada média de 69.945 mensagens por campanha.

Tabela 4. Top 5 Country Codes

CC	# de Mensagens	# de Endereços IP	# de AS's	# de Campanhas
US	3.605.904 (36,93%)	1.406 (20,97%)	302 (17,64%)	233
DE	2.308.181 (23,64%)	175 (2,61%)	53 (3,09%)	33
BA	480.317 (4,92%)	26 (0,38%)	2 (0,11%)	23
CO	388.093 (3,97%)	37 (0,55%)	9 (0,52%)	14
ZA	270.790 (2,77%)	41 (0,61%)	12 (0,70%)	11

4.2. Principais campanhas de *phishing*

Nesta seção apresentaremos as principais campanhas de *phishing* encontradas em nossa base de dados após a separação destas mensagens. Ao todo, foram encontradas 612 campanhas de *phishing*, entretanto, poucas campanhas abrangem a maior parte das mensagens coletadas. Desta forma, descreveremos cada uma das 3 principais campanhas encontradas em nossa base de dados. Estas campanhas foram responsáveis por quase 24% de todas as mensagens classificadas como *phishing*. A Tabela 5 apresenta mais detalhes destas campanhas.

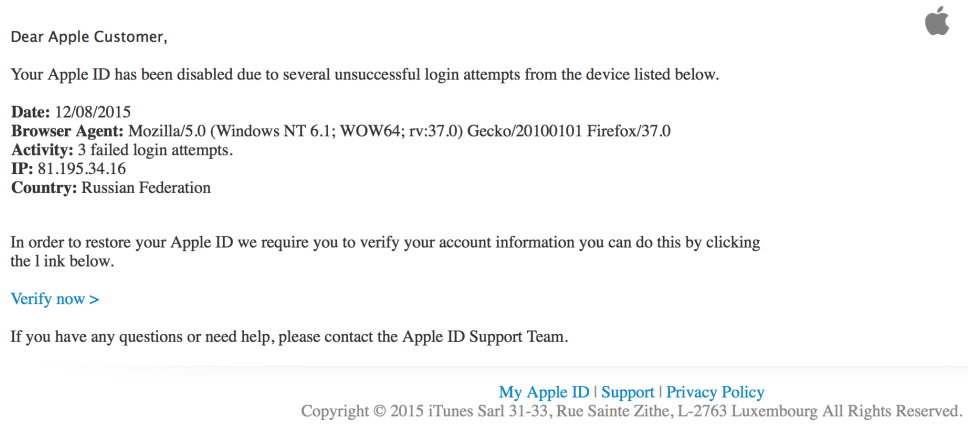
Tabela 5. Top 3 Campanhas

	Características				Categorias					
	Mensagens	**IP**	**AS**	**CC**	**Abordagem**	**Dinheiro**	**Resposta**	**Urgência**	**Formulário**	**Segurança**
C 1	1.124.297	16	5	4	X	X			X	X
C 2	779.359	3	1	1	X			X	X	X
C 3	399.512	30	22	2	X	X	X		X	X

4.2.1. Campanha 1

A maior campanha encontrada em nossa base de dados é composta por 1.124.297 de e-mails, ou seja, 11,52% de todas as mensagens de *phishing* identificadas nos 91 dias de coleta. Como pode ser visto na Figura 3, esta campanha refere-se à um serviço da Apple sendo suspenso e tentando levar a vítima a acessar um link em que será necessário inserir informações a respeito desta conta para reativá-la. Como mostrado na Tabela 5, esta campanha apresenta a catergoria *Abordagem*, uma vez que inicia a mensagem utilizando a saudação *Dear*. As outras categorias presentes são *Dinheiro*, representada pela palavra *Customer*, e *Segurança*, devido as palavras *security* e *account*, além da categoria *Formulário*, pela palavra *information*.

Esta campanha provêm de 16 endereços IP localizados nos seguintes países: Estados Unidos, Alemanha, Colômbia e Grã-Bretanha. Entretanto, apesar dos 16 endereços IP utilizados, 6 destes enviaram baixo número de mensagens, não chagando à 1.000, durante o período analisado. Em contrapartida, o IP 65.29.192.68, listado anteriormente como um dos principais da base, enviou mais de 500 mil *phishing* relacionados a esta campanha.


Dear Apple Customer,

Your Apple ID has been disabled due to several unsuccessful login attempts from the device listed below.

Date: 12/08/2015
Browser Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0
Activity: 3 failed login attempts.
IP: 81.195.34.16
Country: Russian Federation

In order to restore your Apple ID we require you to verify your account information you can do this by clicking the l ink below.

Verify now >

If you have any questions or need help, please contact the Apple ID Support Team.

My Apple ID | Support | Privacy Policy
Copyright © 2015 iTunes Sarl 31-33, Rue Sainte Zithe, L-2763 Luxembourg All Rights Reserved.

Figura 3. Mensagem de exemplo da campanha 1.

4.2.2. Campanha 2

O conteúdo das mensagens da segunda maior campanha, apresentado na Figura 4, refere-se a um banco apresentando um novo processo de autenticação para aumentar a segurança dos seus clientes. Com este pretexto, o atacante induz o usuário a completar informações para utilizar este novo processo, através de um formulário presente acessível por um link presente na mensagem.

Esta campanha também possui alto volume, com aproximadamente 780 mil mensagens, mas que foram enviadas por apenas 3 endereços IP distintos. Estes endereços (212.227.94.138, 212.227.95.8 e 212.227.255.64) se encontram na Alemanha, no AS 8560, e também estão entre os principais endereços IP encontrados em nossa base de dados. Para esta campanha, cada um deles teve uma média de envio de 259 mil *phishings* no período de um mês.

4.2.3. Campanha 3

A terceira campanha de nossa base de dados também refere-se à possível suspensão de uma conta online. No caso do exemplo mostrado, a conta em questão é do banco Wells

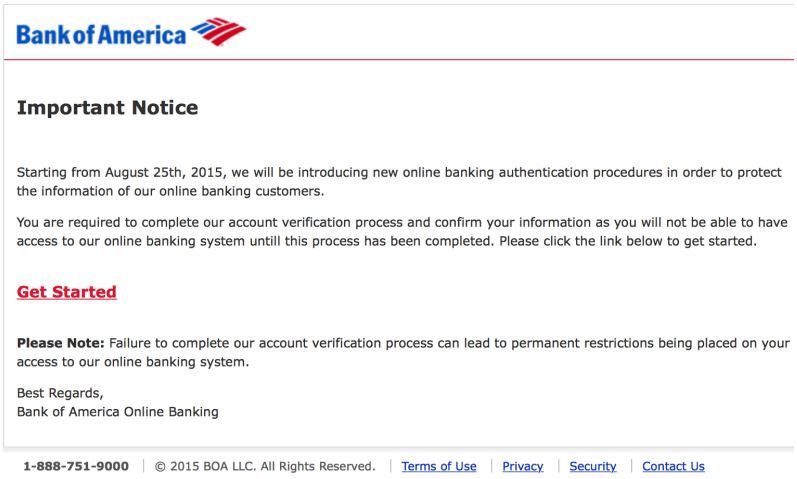


Figura 4. Mensagem de exemplo da campanha 2.

Fargo, entretando, nas demais mensagens desta campanha encontramos referencias a diversos bancos distintos como por exemplo Bank of America e Natwest. Diferente da campanha anterior, que tenta roubar informações do usuário através de um website, acessível por um link presente na mensagem, esta campanha induz o usuário a baixar um documento em anexo, preenchendo-o com informações sensíveis. É possível que o documento em questão represente um *malware*, mas na metodologia do trabalho não cobrimos este tipo de problema. Como trabalho futuro, verificaremos esta questão.

Esta campanha foi enviada por 30 endereços IP distintos presentes nos Estados Unidos e Grã-Bretanha. O fato das mensagens se originarem nestes dois country codes faz sentido, dado que estão tentando abusar de clientes de bancos como Bank of America, localizado nos Estados Unidos e Natwest, originado na Escócia, parte da Grã-Bretanha.

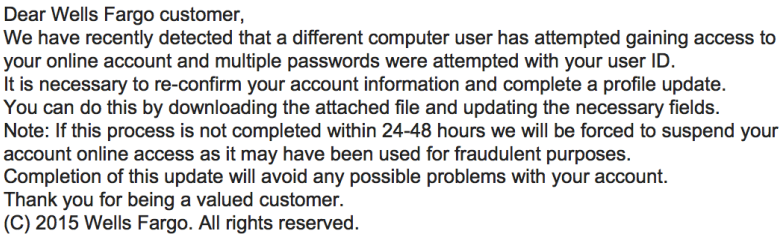


Figura 5. Mensagem de exemplo da campanha 3.

Através destes exemplos, presentes nas principais campanhas identificadas pelo nosso método na base de dados estudada, mostramos a variedade e criatividade dos atacantes na tentativa obter informações sensíveis dos usuários.

4.3. Análise das campanhas de *phishing*

Nesta seção, analisaremos as campanhas de *phishing*. A Tabela 6 apresenta a visão geral das campanhas encontradas. Foram identificadas 612 campanhas distintas. Estas campanhas englobam mais de 8,5 milhões de mensagens, ou seja, mais de 87% de todas as mensagens classificadas como *phishing*. Em média, cada campanha possui 13.984 mensagens, enviadas por apenas 3,16 endereços IP distintos.

Tabela 6. Visão geral das campanhas

Total de Campanhas	612
Média de mensagens	13.984
Média de IPs	3,16
Média de ASes	2,06
Média de CCs	1,82

Como descrito na metodologia, identificamos categorias a serem utilizadas como base para nosso método de detecção. Todas as mensagens identificadas como *phishing* se enquadram em pelo menos uma destas categorias. A Tabela 7 mostra a distribuição das categorias nas campanhas e nas mensagens de *phishing*. A categoria *Abordagem* é aquela aquela que possui maior número de campanhas utilizando-a. Do total de 612 campanhas, 480, com cerca de 7,4 milhões de e-mails, utilizaram alguma forma de abordagem ao usuário. Como dito anteriormente, o objetivo do atacante é ludibriar o usuário e levá-lo a crer que a mensagem é legítima, logo, abordar cordialmente os usuários é um passo fundamental para atingir tal meta. A segunda categoria mais presente nas mensagens da base avaliada é *Dinheiro*. Como descrevemos anteriormente, uma forma fácil de chamar a atenção da vítima é prometendo-a dinheiro fácil. Portanto, os atacantes utilizam muito desta artimanha na tentativa de conseguir informações da vítima. Outra forma utilizada pelos *phishers* para roubar informações dos usuários é fazendo com que ele preencha um formulário com seus dados. Os mais diversos motivos são apresentados pelos atacantes para tentar fazer com que a vítima o faça, como por exemplo a necessidade de atualização de dados cadastrais para que a conta de determinado serviço não seja banida. Com isso, a terceira categoria mais presente foi de *Formulário*. Encontramos 388 campanhas, responsáveis pelo envio de mais de 6 milhões de mensagens apresentando tal comportamento, mostrando que esta é uma categoria importante utilizada nestes tipos de ataque.

Tabela 7. Categoria das campanhas

	Abordagem	**Dinheiro**	**Resposta**	**Urgência**	**Formulário**	**Segurança**	**Total**
Campanhas	480	373	318	236	388	308	612
Mensagens	7.412.701	6.254.321	3.552.518	3.701.086	6.099.531	5.785.346	8.558.237

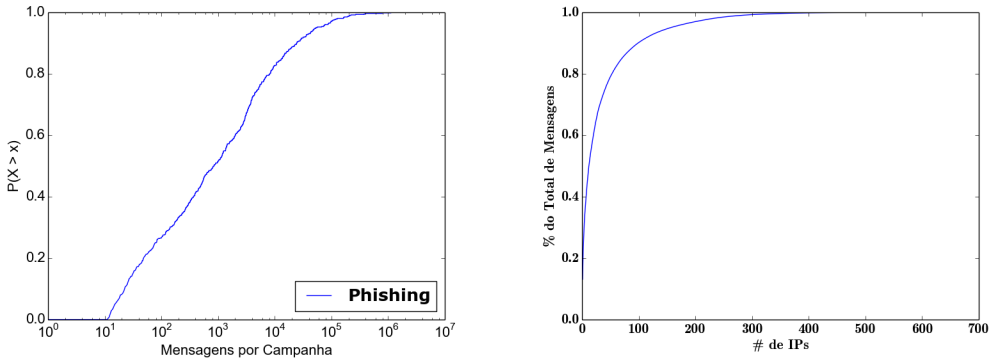
Considerando os conjuntos de categorias às quais as mensagens pertencem, que são 57 de 63 possíveis, podemos afirmar que a nossa metodologia é versátil e capaz de identificar as várias modalidades de phishing. Na Tabela 8, são apresentados os 5 conjuntos que envolveram o maior número de mensagens. O principal conjunto, englobando mais de 1,8 milhão de mensagens, quase 50% do total de *phishing*, envolve *Abordagem*, *Dinheiro*, *Formulário* e *Segurança*. Estas categorias foram encontradas no exemplo da maior campanha encontrada (Figura 3). Além disso, as categorias *Abordagem* e *Formulário* estão presentes em todas estas mensagens, se mostrando características marcantes de mensagens de *phishing*. Uma maneira de evitar estes ataques, como mostrado em [Sheng et al. 2007], é treinar e ensinar aos usuários as diferentes formas de ataque utilizada pelos *phishers*, minimizando a possibilidade de que estes se tornem vítima. Logo, um viés a ser utilizado nas técnicas de ensino aos usuários é apresentar estas características tão importantes na construção das mensagens do *phishing*, como mostrado através dos resultados presentes na Tabela 7.

Quanto ao volume de mensagens das campanhas, podemos observar, através do

Tabela 8. Conjuntos de categorias

Conjuntos de categorias						Características				
Abordagem	**Dinheiro**	**Resposta**	**Urgência**	**Formulário**	**Segurança**	**Mensagens**	**Camp.**	**IP**	**AS**	**CC**
X	X			X	X	1.834.929	33	104	68	31
X	X	X		X	X	839.739	40	94	60	28
X			X	X	X	819.663	8	14	10	14
X	X	X	X	X		555.474	35	85	52	29
X	X	X	X	X	X	511.723	60	172	95	36

gráfico presente na Figura 6(a), que cerca de 50% das campanhas possuem 1.000 mensagens ou menos e 10% das mensagens, ou seja, apenas 61 campanhas tem mais de 24 mil e-mails. Vale ressaltar que, como descrito na metodologia, não consideramos campanhas que possuem 10 mensagens ou menos. Com relação à concentração das mensagens na campanhas, pode ser visto, no gráfico presente na Figura 6(b), que quase 90% das mensagens se encontram nas 100 maiores campanhas. Como dito anteriormente, uma melhoria no combate ao *phishing* é focar os recursos nos principais endereços IP e principais campanhas, mitigando significativamente o tráfego deste tipo de mensagem pela Internet.



(a) CDF do número de mensagens por campanhas.

(b) CDF da concentração de mensagens por campanha

Figura 6. CDF's das campanhas

5. Conclusão e Trabalhos Futuros

Neste trabalho apresentamos um método adaptativo para identificação de mensagens de *phishing* que utiliza categorias bem definidas na literatura e expande-as, através da técnica Word2Vec, para identificar possíveis nuances presentes nas diferentes mensagens de *phishing* ao longo do tempo. Utilizando o método proposto conseguimos identificar, com taxa de acerto de aproximadamente 95%, mais de 9,7 milhões de e-mails de *phishing*, quantidade bastante relevante, principalmente quando comparada à quantidade utilizada por outros trabalhos na literatura.

Além disso, mostramos características do *phishing*, como por exemplo, o fato dele ser enviado quase somente através do protocolo SMTP. Mostramos ainda que alguns poucos endereços de origem, comumente localizados nos Estados Unidos e Alemanha, são responsáveis pela maior parte deste tráfego, e que combatendo-os, tal problema seria significativamente mitigado. Através da geração de campanhas, mostramos que poucas, normalmente ligadas a bancos, respondem por um percentual elevado do total de mensagens

estudados. Como trabalhos futuros, pretendemos aprimorar a técnica de identificação de *phishing*, utilizando outras métricas como a forma de envio. Além disso, pretendemos expandir as análises realizadas, estudando também anexos das mensagens além das URL's que estas possuem.

Agradecimentos

Este trabalho foi parcialmente financiado por NIC.BR Fapemig, CAPES, CNPq, e pelos projetos MCT/CNPq-InWeb (573871/2008-6), FAPEMIG-PRONEX-MASWeb (APQ-01400-14), e H2020-EUB-2015 EUBra-BIGSEA (EU GA 690116, MCT/RNP/CETIC/Brazil 0650/04).

Referências

Aggarwal, S., Kumar, V., e Sudarsan, S. D. (2014). Identification and detection of phishing emails using natural language processing techniques. Em *Proc. of the 7th Int'l Conference on Security of Information and Networks*, New York, USA. ACM.

Baeza-Yates, R. A. e Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Bergholz, A., De Beer, J., Glahn, S., Moens, M.-F., Paaß, G., e Strobel, S. (2010). New filtering approaches for phishing email. *J. Comput. Secur.*

Brown, C. D. e Davis, H. T. (2006). Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 80(1):24 – 38.

Fette, I., Sadeh, N., e Tomasic, A. (2007). Learning to detect phishing emails. Em *Proceedings of the 16th International Conference on World Wide Web*, WWW '07.

Kumaraguru, P., Rhee, Y., Acquisti, A., Cranor, L. F., Hong, J., e Nunge, E. (2007). Protecting people from phishing: The design and evaluation of an embedded training email system. Em *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

Las-Casas, P. H. B., Guedes, D., Jr., W. M., Hoepers, C., Steding-Jessen, K., Chaves, M. H. P., Fonseca, O., Fazzion, E., e Moreira, R. E. A. (2013). Análise do tráfego de spam coletado ao redor do mundo. Em *Anais do simpósio brasileiro de redes de computadores e sistemas distribuídos (SBRC)*. SBC.

Marchal, S., François, J., State, R., e Engel, T. (2014). Phishscore: Hacking phishers' minds. Em *International Conference on Network and Service Management (CNSM)*, p 46–54.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., e Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Sheng, S., Magnien, B., Kumaraguru, P., Acquisti, A., Cranor, L. F., Hong, J., e Nunge, E. (2007). Anti-phishing phil: The design and evaluation of a game that teaches people not to fall for phish. Em *Proc. of the 3rd Symp. on Usable Privacy and Security*, SOUPS '07, p 88–99, New York, NY, USA. ACM.

Sheng, S., Wardman, B., Warner, G., Cranor, L. F., Hong, J., e Zhang, C. (2009). An Empirical Analysis of Phishing Blacklists. Em *Conference on Email and Anti-Spam*.

Steding-jessen, K., Vijaykumar, N. L., e Montes, A. (2007). Using low-interaction honeypots to study the abuse of open proxies to send spam.

Xiang, G., Hong, J., Rose, C. P., e Cranor, L. (2011). Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Trans. Inf. Syst. Secur.*, 14(2):21:1–21:28.

Zhang, Y., Hong, J. I., e Cranor, L. F. (2007). Cantina: A content-based approach to detecting phishing web sites. Em *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, p 639–648, New York, NY, USA. ACM.

A Selective Defense for Mitigating Coordinated Call Attacks

Marcilio O. O. Lemos[1], Yuri Gil Dantas[2], Iguatemi E. Fonseca[1], Vivek Nigam[1] and Gustavo Sampaio[1]

[1]Federal University of Paraíba, Brazil

[2]Technische Universität Darmstadt, Germany

{marcilio.cc.lemos,gbritosampaio}@gmail.com,

{iguatemi,vivek}@ci.ufpb.br,

dantas@mais.informatik.tu-darmstadt.de

Abstract. *Telephony Denial of Service (TDoS) attack is a form of Denial of Service (DoS) attack that targets telephone services, such as Voice over IP (VoIP), not allowing legitimate users to make calls. This paper proposes a new selective defense for mitigating a type of TDoS called Coordinated Call Attack where attackers simply call to each other exhausting the target VoIP server's resources and denying service to legitimate users. Our defense builds on the defense SeVen for mitigating Application Layer DDoS attacks. We implemented and integrated SeVen in usual VoIP systems using the SIP protocol and carried out a number of experiments: without SeVen, less than 15% of users can access the target VoIP service, whereas with SeVen, around 90% of users can access the same service.*

1. Introduction

Distributed Denial of Service (DDoS) attacks have been a great concern to network administrators since the origins of the Internet targeting all sorts of services. Voice over IP has been widely used for audio and video communications due to its low costs and acceptable quality. As such, it has been target of DDoS attacks such as VoIP amplification attacks [30, 22] and the SIP flooding attack [30]. Telephony Denial of Service (TDoS) attacks are DoS attacks that target telephone services, such as VoIP, not allowing legitimate users to make calls. TDoS attacks have been reported targeting hospital systems [2, 7] and systems for emergency lines (like the American 911 system) [6]. Moreover, according the FBI, 200 TDoS attacks were identified only in 2013 [7].

Available defenses for DDoS targeting VoIP services, such as [27, 17, 29, 31], are constructed to mitigate attacks by analysing traffic flows and whenever there is an unusual increase of traffic, suitable mechanisms are placed, such as blocking IPs[1]. A recent class of DDoS attacks, called Application-Layer DDoS Attacks (ADDoS), are able to bypass such defense mechanisms because they are carried out by generating traffic that is similar to usual client traffic. Examples of ADDoS attacks include the Slowloris [23], POST [20] and Slowread [24] attacks exploiting the HTTP protocol, and the Coordinated Call Attack [11] exploiting the SIP protocol used by VoIP applications.

The Coordinated Call Attack [11] exploits the fact that pairs of attackers, Alice and Bob, can collude to exhaust the resources of the VoIP server. Assume that Alice and

[1]see also http://itsecurity.telelink.com/tdos-attacks/ accessed on March 2016

Bob are valid registered users. This can be easily done for many VoIP services. The attack goes by Alice simply calling Bob and trying to stay in the call as long as she can. Since the server allocates resources for each call, by using enough pairs of attackers, they can exhaust the resources of the server and deny service to legitimate clients. This is a simple, but ingenious attack, as only a small number of attackers is needed generating a small network traffic (when compared to SIP flooding attack for example), being, thus, hard for network administrators to detect and counter-measure such attack.

Our recent work [16] proposed the use of selective strategies for mitigating (HTTP) Application-Layer DDoS attacks in the form of the tool called SeVen. Our defense mechanism is governed by probability functions that specify the chances of a request to be dropped whenever the service is overloaded. We showed in our previous work that by using simple uniform distributions, SeVen can be used against a number of attacks exploiting the HTTP protocols (Slowloris and POST). However, that work only considered ADDoS attacks exploiting the HTTP protocol against web-servers.

This paper investigates the use of selective strategies for mitigating TDoS attacks, in particular, the Coordinated Call Attack. Our contributions are three-fold:

1. **A New Selective Strategy for VoIP Services:** We propose a new selective strategy suitable for VoIP services. While in our previous work [16], we used simple uniform probabilities, this paper uses more sophisticated definitions for deciding which calls that are going to be selected to continue and which should be dropped;
2. **Integration of SeVen with VoIP Servers:** We investigated how SeVen can be used in conjunction with typical VoIP servers, in particular, Asterisk [1]. We implemented our solution using SeVen as a proxy for VoIP server;
3. **Experimental Results:** Finally, we carried out a number of experiments showing that our solution can mitigate Coordinated Call Attacks. When under attack and without using our defense, we observed that only 15% of users could access the VoIP service. All of these clients were able to complete their calls. When using SeVen, we observed that around 90% of clients could access the VoIP service. Of these, 63% were able to complete their calls. The remaining 27% were interrupted by SeVen being able to stay using the VoIP service in average 60% of the intended call duration.

We start in Section 2 explaining the SIP protocol used by VoIP services and the Coordinated Call Attack. Section 3 introduces the new selective strategy used to mitigate the Coordinated Call Attack. In Section 4, we describe the experimental set-up used to validate our defense mechanism as well as the quality measures used. We also show our main findings which validate the efficiency of our defense mechanism. Finally in Section 5, we discuss related work and conclude pointing to future work.

2. VoIP and the Coordinated Call Attack

Nowadays the most widely used signaling protocol in VoIP communication is the Session Initiation Protocol (SIP) [25]. Figure 1 depicts the messages that are exchanged during a call. In the initiation phase, Alice, the caller, sends an INVITE message to the SIP Proxy, such as Asterisk [1] server, with the details of Bob, the party Alice wants to talk with. The proxy searches for Bob's details (such as his IP) and sends an INVITE message to

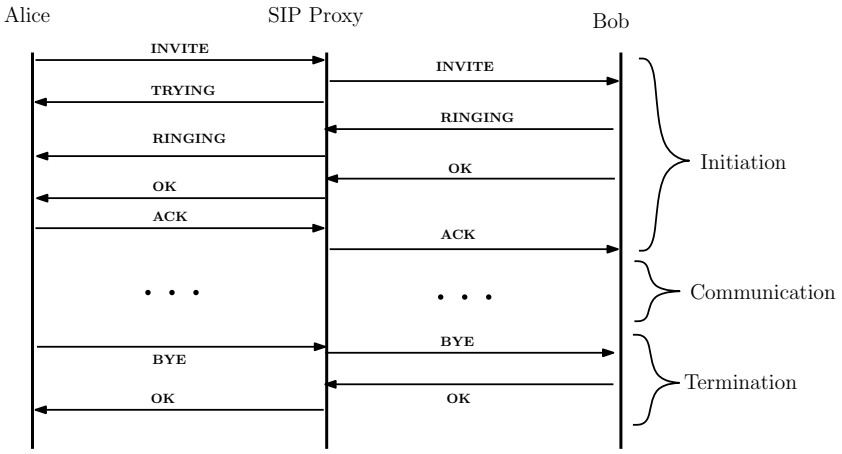


Figure 1. Session Initiation Protocol (SIP)

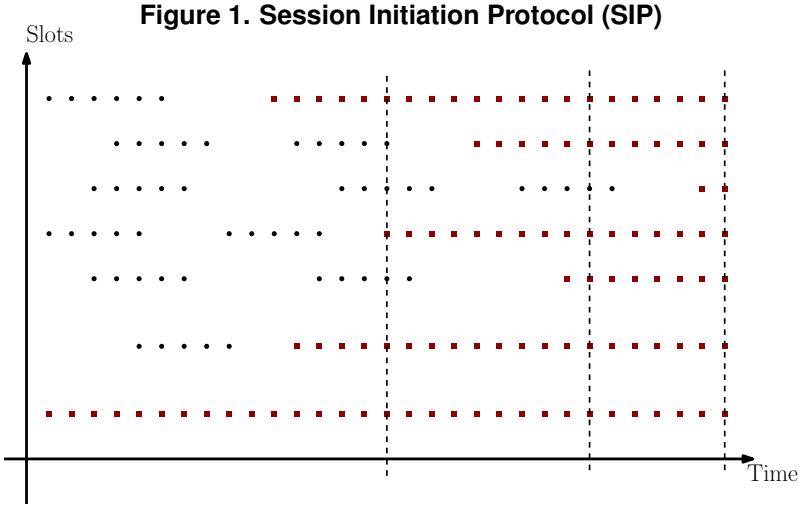


Figure 2. Illustration of the Coordinated VoIP Attack. (Black) Circles represent legitimate calls, while (red) squares represents an attacker call.

Bob with Alice's identification.[2] At the same time, the SIP Proxy sends Alice a TRYING message indicating that it is checking whether Bob is available. If he is available, Alice and Bob exchange through the proxy a number of messages (RINGING, OK, ACK) to establish the connection. At this point, the communication phase starts when Alice and Bob can talk or exchange data. When the call is terminated by one of the parties (in Figure 1, Alice), then a BYE message is sent to the proxy which forwards it to Bob who responds with an OK message acknowledging the end of the call.

2.1. Coordinated Call Attack

A pair of colluding attackers, A_1 and A_2, that are registered in the VoIP service,[3] call each other and stay in the call for as much time as they can. The attackers exchange with the SIP server the same initiation messages as in a normal SIP session as shown in Figure 1. Once the call is established, the attackers stay in the call for indefinite time. They may be disconnected by some Timeout mechanism establishing (very large) time bounds on call duration. During the time that A_1 and A_2 are communicating, they are using resources of

[2]In practice, Asterisk reformulates the data send by Alice according to its own user database adding further information such as its own identification tags.

[3]Or alternatively two honest users that have been infected to be zombies by some attacker.

the server. As usual, VoIP servers have an upper-bound on the number of simultaneous calls they can handle. By using enough colluding attackers, the target VoIP server will reach its limit denying new (legitimate) calls to be established.

Figure 2 illustrates why this attack is so effective. The attacker can slowly occupy the call slots available in the SIP Server. Since they stay indefinite time, once the slot is occupied it is never made free (without a defense mechanism) and therefore eventually the attacker will be using all slots not allowing legitimate clients to use the VoIP service.

An important difference of this attack to other DDoS on VoIP services, such as SIP-flooding, is the fact that the attacker's traffic is not significantly different to usual legitimate traffic. Firstly, the attackers follow correctly the protocol sending the expected messages as specified by the SIP protocol. Secondly, the attack does not have to generate a large number of calls at a particular instance. It can slowly populate the VoIP resources with traffic load, *i.e.*, call rate, similar to client call rate. The overall traffic would correspond to expected traffic load and therefore not activating any defense mechanism.

3. SeVen

Our defense, SeVen, uses a selective strategy [16]. In a nut-shell, *whenever the application is overloaded*, that is, its capacity is full, and a new request R arrives, SeVen acts as follows:

1. SeVen decides (using some probability distribution P_1) whether the application should process R or not;
2. If SeVen decides not to process R, then it simply returns a message to the requesting user that the service is not available;
3. If SeVen decides to process R, then as the application is overloaded, it should decide (using some probability distribution P_2) which request currently being processed should be dropped. This decision is governed by P_2, a distribution probability *which may depend on the state of the existing request*.

Intuitively, this strategy works because when the application is overloaded, it is very likely that it is suffering a DDoS attack. Therefore, whenever a new request arrives and SeVen decides to process it, the probability of dropping an attacker is much higher. Figure 2 illustrates this situation: the vertical dashed lines indicate some moments when the application is overloaded and new requests arrive. The number of square dots, representing attackers, is much greater than the number of circle dots, representing legitimate clients, when the attack is being carried out. While without SeVen the application would simply deny service to all requesting clients, SeVen allows new clients to be served.

The first main contribution of this paper is on the definition of the probability distributions P_1 and P_2, in particular, the probability P_2. While in our previous work [16] P_2 was uniform which was suitable for Web-Servers, we need a more elaborate distribution. P_2 will depend on (1) the status of the request and (2) on the duration of a call.

We consider two types of status for requests:

- WAITING: A request is WAITING if it did not yet completed the initiation phase of the SIP protocol (see Figure 1). That is, it is still waiting for the responder to join the call and start to communicate;
- INCALL: A request is INCALL if the initiation phase of the SIP protocol have been completed and the initiator and the responder are already communicating (or simply

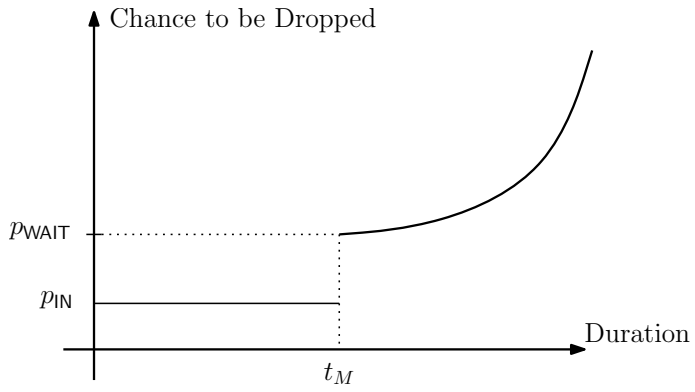


Figure 3. Graph (not in scale) illustrating the behavior of SeVen according to the status of a request and the duration of INCALL request. p_{WAIT} is the factor for dropping a WAITING request, while p_{IN} the factor for dropping a INCALL request.

in a call).

Thus, any incoming INVITE request assumes the status of WAITING, and it can change its status to INCALL once the initiation part of SIP is completed.

We assume here that it is preferable to a VoIP server, when overloaded, to drop WAITING requests than INCALL requests that are communicating not for a *very long duration*. In many cases, it is true that interrupting an existing call is considered to be more damaging to server's reputation than not allowing a user to start a new call. This could also be modeled by configuring the probability distributions of SeVen accordingly. Moreover, to determine whether a call is taking too long, we assume that the server knows what is the average duration, t_M, of calls.[4]

The chance of an INCALL request to be dropped increases exponentially. This function was based on using a Poisson distribution[5] once this has a duration of more than t_M. Figure 3 depicts roughly how the chance of dropping a call increases with the call duration. The actual equation is of the form, where t is the call duration:

$$d(t) = \begin{cases} p_{\mathsf{WAIT}} & \text{if } t = 0 \\ p_{\mathsf{IN}} & \text{if } 0 \leq t \leq t_M \\ p_{\mathsf{WAIT}} + e^{\alpha t / t_M} & \text{if } t > t_M \end{cases} \quad (1)$$

We apply this function to the roulette-wheel selection method [19] in order to produce the probability distribution P_2 that decides which request currently being processed should be dropped. This function is used because we think it is reasonable for many applications. Of course, there are many decision options for these probabilities which will depend on the intended application. For instance, one could consider that the Poisson distribution should begin only a period after t_M, or that it should be another distribution, etc. It will depend on the specific requirements of the defense.

Finally, we point out that we also formally specified this strategy in the computational tool Maude [13] and used advanced model checking techniques, namely Statistical Model-Checking [21], to have a first impression of whether this defense would work or

[4]The value of t_M can be obtained by the history of a VoIP provider's usage.

[5]We used a Poisson distribution because such distributions are normally used for modeling telephone calls arrival [12]

not. As our model checking results (which are detailed in another work [15]) were satisfactory, we implemented the defense and carried out experiments, which will be detailed in Section 4.

3.1. SeVen by Example

The formal specification of SeVen is detailed in our previous work [16]. We illustrate how it works with an example referring the details of the mechanism to [16].

Assume for this example that the VoIP server can only handle 3 simultaneous calls. Moreover that the average time of calls is $t_M = 5$ (just for illustration) and that currently, the server is processing two call requests identified as id_1 and id_2 as follows:

$$\mathcal{P}_0 = [\langle id_1, \mathsf{INCALL}, 8\rangle, \langle id_2, \mathsf{WAITING}, 0\rangle]$$

where $\langle id, st, dur\rangle$ specifies that the call id has status st and dur is the duration of the call, being 0 whenever $st = \mathsf{WAITING}$. Thus, id_1 is calling for 8 time units, while id_2 is still waiting for the responder.

Assume that a new request arrives, identified as id_3. Since the server's capacity is not reached, it accepts this request updating the status of the server to the following reaching its maximum capacity:

$$\mathcal{P}_1 = [\langle id_1, \mathsf{INCALL}, 8\rangle, \langle id_2, \mathsf{WAITING}, 0\rangle, \langle id_3, \mathsf{WAITING}, 0\rangle]$$

Assume that one time unit passes and that the id_2 completes the initiation phase. The status of the server is updated to:

$$\mathcal{P}_2 = [\langle id_1, \mathsf{INCALL}, 9\rangle, \langle id_2, \mathsf{INCALL}, 1\rangle, \langle id_3, \mathsf{WAITING}, 0\rangle]$$

Now consider that a new request id_4 arrives. Since the server is in its maximum capacity, SeVen should decide whether it processes id_4 or not. SeVen throws a coin (according to the probability distribution P_1 which is not important for this paper. You can safely assume a fair coin). If it returns 0, then it does not process id_4; otherwise it does process id_4.

Assume that SeVen decides to process id_4. SeVen now has to decide which one of the requests currently being processed, id_1, id_2 or id_3, should be dropped. It chooses one using the probability distribution generated from function depicted in Figure 3. Thus, id_1 has a greater chance of being chosen than id_2 and id_3 because its call has a duration greater than t_M ($9 > 5$). Moreover, id_3 has a greater chance of being chosen than id_2 as $p_{\mathsf{IN}} < p_{\mathsf{WAITING}}$.

Assume that SeVen decides to drop id_1. The new status of the server will be:

$$\mathcal{P}_3 = [\langle id_2, \mathsf{INCALL}, 1\rangle, \langle id_3, \mathsf{WAITING}, 0\rangle, \langle id_4, \mathsf{WAITING}, 0\rangle]$$

Finally, say that id_2 finishes its call. The resulting server status will be:

$$\mathcal{P}_4 = [\langle id_3, \mathsf{WAITING}, 0\rangle, \langle id_4, \mathsf{WAITING}, 0\rangle]$$

where the server is no longer overloaded.

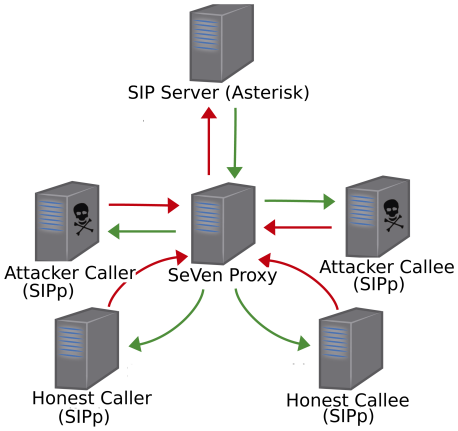


Figure 4. Experiment Topology

In practice, the values for $p_{\mathsf{IN}}, p_{\mathsf{WAIT}}$, the capacity of the server and the t_M will depend in the application. Moreover, the duration of call would be measured in seconds (or even milliseconds).

Remark: Other parameters besides call duration could be used in configuring the chances to drop clients. Dantas's Master thesis [14] details several other options, such as different classes of users (*e.g.*, Gold, Silver and Bronze), the type of connections, etc. Their use will depend on the particular situation. We leave the task to incorporate other parameters to future work.

4. Experimental Results

4.1. Set-Up

In our experiments, we used Asterisk version 13.6.0 which is a SIP server widely used by small and mid size companies for implementing their VoIP services. We assume there are honest users and malicious attackers which try to make the VoIP unavailable. Both the traffic of the honest users and the attackers are emulated using the tool SIPp [5] version 3.4.1. SIPp generates calls which may be configured as the caller or the callee. Thus, in our experiments, we used pairs of SIPp, one pair for generating the honest user calls and the other pair for generating the attacker calls. Finally, we developed the SeVen proxy in C++ which implements the selective strategy described in Section 3.1.

Figure 4 illustrates the topology of the experiments we carried out. To make a call, the pairs of SIPp send messages to the SeVen proxy which on the other hand forwards them to Asterisk. Similarly, any message generated by Asterisk is forward to the SeVen proxy which then forwards them to the corresponding users. Therefore, SeVen is acting as an *Outbound Proxy* for both Asterisk and the pairs of SIPp. For our experiments, it is enough to use a single machine. We used a machine with configuration Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz and 8 GB de RAM.

Finally, we set the duration of the calls generated by SIPp as follows:

- **Total Call Duration of Clients:** Whenever we generate a new call, we generate randomly with a uniform probability a natural number between $[1, 2 \times t_M]$. Thus legitimate user calls may at most take twice the average call duration. SIPp ends the call when its corresponding call duration is reached;

- **Call Duration of Attackers:** Following the Coordinated Call Attack, we do not limit the call duration of an attacker call. He call communicate for indefinite time.

Parameters We use the following parameters to configure our experiments:

- **Average Call Duration (t_M)** – We assume known what is the average duration of calls. This can be determined in practice by analyzing the history of calls. We assume in our experiments that $t_M = 5$ minutes;
- **Dropping Factor** – We assume the following values for the dropping chance function (Eq 1):
  - $p_{\mathsf{IN}} = 2$;
  - $p_{\mathsf{WAIT}} = 8$;
  - $\alpha = 1.89$.

  These values were chosen so that the probability of dropping increases in a reasonable fashion after the call duration is greater than t_M. Sample values are as follows, recalling that $t_M = 5$:

Call Duration (mins)	Dropping Factor
6	12.37
8	17.31
10	27.84

  That is, the chance of dropping a call with duration of 10 minutes is approximately 3 times greater than dropping a call whose status is WAITING (27.84/8). This is a reasonable ratio. However, according to the specific application other values can be set for $p_{\mathsf{IN}}, p_{\mathsf{WAIT}}$ and α. Finally, the choice of setting $p_{\mathsf{WAIT}} = 4 \times p_{\mathsf{IN}}$ was selected so that the calls with duration less than t_M have much less chance of being dropped than the calls that are still waiting for the responder.
- **SIP Sever Capacity (k)** – This is the number of simultaneous calls the SIP server can handle. We set $k = 50$ which is a realistic capacity for a small company allowing 100 users (2×50) to use the service at the same time.
- **Experiment Total Time (T)** – Each one of our experiments had a duration of 60 mins, that is, 12 times the average call of clients. With this duration, it was already possible to witness the damage caused by the Coordinated Call Attack as well as the efficiency of our solution for mitigating this attack.
- **Traffic Rate (R)** – Using a server with capacity of 50, we calculated using standard techniques [10] what would be a typical traffic of such a server. It is $R = 9.9$ calls per minute. In our experiments, we split this rate among clients and attackers. Recall that the Coordinated Call Attack does not generate traffic different from the expected traffic so that it can bypass usual defenses based on network traffic analysis. Thus, the total traffic in our experiments is always less or equal than R.

Quality Measures For our experiments, we used the following three quality measures for our calls:

- **Complete Call:** A call is complete whenever its status changed from WAITING to INCALL and *it is able to stay in status* INCALL *for its corresponding call duration*.

That is, the caller was able to communicate with the responder for all the prescribed duration;

- **Incomplete Call:** A call is incomplete whenever its status changed from WAITING to INCALL, but *it was not able to stay in status* INCALL *for its corresponding call duration*. That is, the caller was interrupted before completing the call;
- **Unsuccessful Call:** A call is unsuccessful if it did not even change its status from WAITING to INCALL. That is, the caller did not even have the chance to speak with the responder.

Intuitively, complete calls are better than incomplete calls which are better than unsuccessful calls. In order to support this claim, we also computed the average duration call of the incompleted calls, that is, the time that users in average where able to stay communicating before they were interrupted by SeVen.

4.2. Experimental Results

We carried out a number of experiments analyzing the attack and the effectiveness of our defense. Three types of experiments were carried out:

- **Type 1:** Attacking the Asterisk server without using SeVen to defend it;
- **Type 2:** Attacking the Asterisk server and using SeVen to defend it;
- **Type 3:** Not attacking the Asterisk sever but still using SeVen.

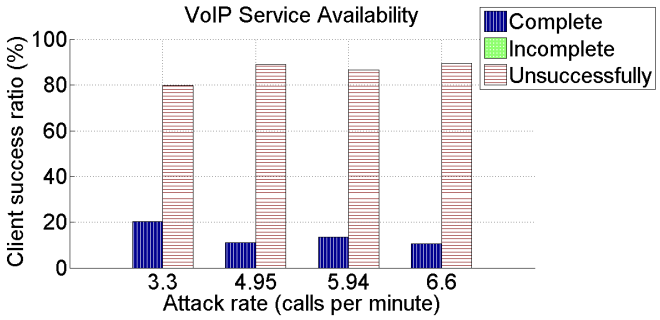
Figure 5 depicts the availability results for legitimate clients for each type of experiment. In our experiments with attack, we varied the attacker call-rate from 3.3 calls per minute until 6.6 calls per minute. The client call rate was adjusted in order to preserve the total rate of 9.9 calls per minute which is a typical call rate for the Asterisk server as detailed in Section 4.1. Thus for the scenario with an attacker call rate of 3.3 calls per minute, the client call rate was 9.9 - 3.3 = 6.6 calls per minute, which corresponds to a ratio of 2 client calls for each attacker call. Similarly, when the attacker call rate used is 6.6 calls per minute, we generated 3.3 calls per minute for legitimate clients, which corresponds to a ratio of 2 attacker calls for each client call.

When under attack and not running SeVen (Figure 5(a)), we observed that the ratio of unsuccessful calls was very high (80-90% of total client calls). This means that these clients were not even able to start a call. The remaining calls (10-20% of total client calls) were all successful, that is, each one could stay using the VoIP service for the total time assigned to them.

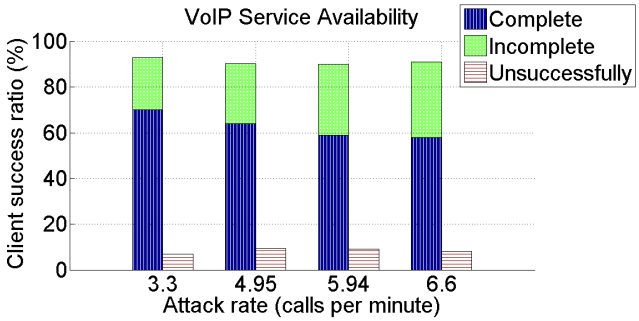
On the other hand, when under attack and using SeVen (Figure 5(b)), we observed that clients had many more successful calls (60-70% of total client calls), while some (0-10% of total client calls) were not able to even start the call. The remaining calls (20-30% of total client calls) were interrupted while calling due to SeVen's DDoS mitigation strategy. Therefore, around 90% of the clients were able to use (even if some in an incomplete fashion) the VoIP service.

Finally, we observed that SeVen did not affect the client success ratio when Asterisk is not under attack as depicted in Figure 5(c), where all client calls were successful.

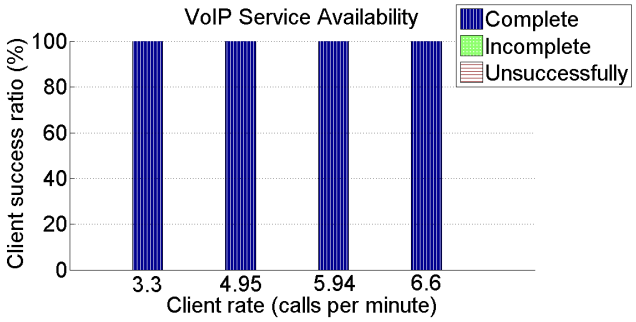
In order to understand better the profile of the incomplete calls in the scenario with attack and using SeVen, we measured the following ratio for each incomplete call: $\frac{t_D}{t_T}$, where t_D is the duration call, that is, the time when the call started until it was interrupted,



(a) Client Success when under attack and *not* running SeVen.



(b) Client Success when under attack and running SeVen.



(c) Client Success when not under attack and running SeVen.

Figure 5. Availability Results

and t_T is the intended total call duration time. Figure 6 depicts the average call duration ratio of the incomplete calls. The average was around 60% for each experiment with different attacker call rates. This means that the clients in incomplete calls still were able to communicate for more than half of the corresponding duration of the call. This supports our claim that incomplete calls are indeed much better than unsuccessful calls in which clients are not even able to start the call.

Finally, in order to understand better the behavior of the Coordinated Call Attack and in particular, how effective our defense is, we studied how many calls were occupied by attackers over time. Figure 7 depicts the results for the case when the attacker call rate is 4.95 calls per minute, that is, the same rate as the client call rate (9.9 - 4.95 =

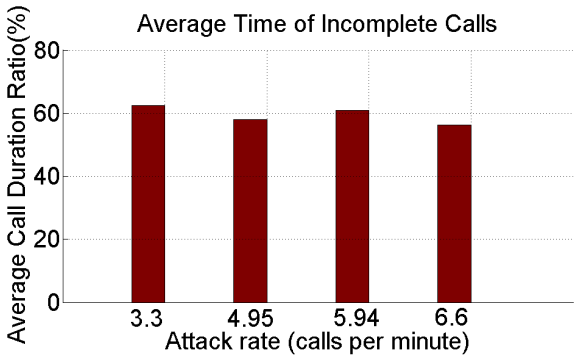
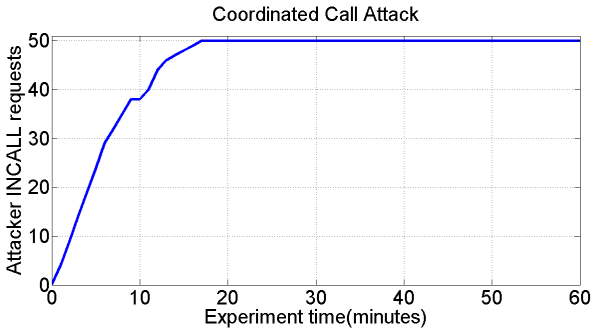
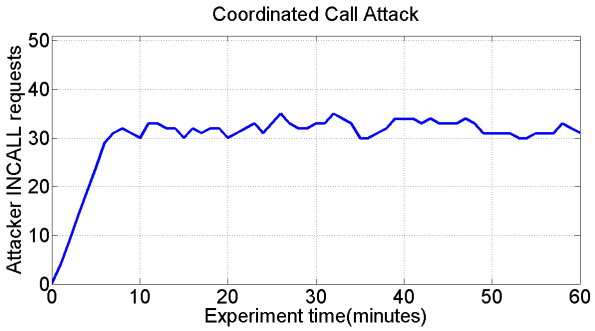


Figure 6. Average Duration of Incomplete Calls



(a) Without using SeVen.



(b) Using SeVen.

Figure 7. Attacker Occupancy over time when Attacker Call rate is 4.95 calls per minute.

4.95 calls per minute). The cases with other attacker call rates had similar results. We observed that when not using SeVen (Figure 7(a)), the attacker is able to occupy all the 50 call positions after 17 minutes. After this point, no legitimate client is able to use the VoIP service. This means that if we considered even longer experiments, the availability of clients (Figure 5(a)) would tend to zero as the clients are only able to access the VoIP service in the first 17 minutes of the experiment. *Indeed, we carried out experiments first initializing the attackers letting them occupy all available slots of the VoIP server and only then initialized the clients. The result of this experiment was 100% of unsuccessful calls.*

On the other hand, with SeVen (Figure 7(b)), we observe that the attackers were

not able to occupy all positions of the VoIP server. After 8 minutes, the attackers were able to occupy a bit more than 30 slots, leaving the remaining slots available to legitimate clients. This remained so for the rest of the experiment. It demonstrates that the selective strategy indeed works as intuitively described in Section 3.

5. Conclusions, Related and Future Work

This paper introduced a new selective strategy for mitigating the Coordinated Call Attack on VoIP services. We implemented this defense and carried out a number of experiments and analyzes using an off-the-shelf VoIP server (Asterisk) demonstrating the effectiveness of our defense. Under an attack and using our defense, around 90% of legitimate clients were able to access the VoIP services, 63% of which used the service until full satisfaction. In contrast, without our defense, only 15% of clients had access to the VoIP service.

Most of the existing work [18, 28, 26, 27, 29, 31] on mitigating DoS attacks on VoIP services focuses on Flooding attacks, such as SIP-Flooding attack. They analyze the network traffic and whenever they observe an abrupt increase in the traffic load, they activate their defenses. The network traffic is usually modeled using some statistical approach, such as correlating the number of INVITE requests and the number of requests that completed the SIP initiation phase [18] or using more complicated metrics such as helling distance to monitor traffic probability distributions [28, 26, 27]. Other solutions place a lower priority on INVITE messages, which are only processed when there are no other types of request to be processed [29, 31].

As the Coordinated Call Attack emulates legitimate client traffic not causing an unexpected sudden increase in traffic, all these defenses are not effective in mitigating the Coordinated Call Attack. The few solutions we found in the literature for this type of attack are commercial tools that act as a firewall which monitors all the call traffic and the signaling [4, 9] or analyze audio samples [3] in order to differentiate the fraudulent calls from the legitimate ones. Less sophisticated mechanisms [8] monitors all the incoming requests and rejects those whose IPs do not belong to a list of trusted IPs. Clearly such approaches does not work well when the attackers are malicious users who's IPs are in the trusted list and are not using automation to make the calls. In addition, these commercial tools can be expensive for small businesses to purchase and maintain, and they do require technical expertise for proper installation.

One main advantage of our proposed solution is that it is not tailored using many specific assumptions on type of service. The only assumption used is a previous knowledge of the average call duration, which can be easily inferred from the service call history. Moreover, our solution can be easily integrated with other mechanisms such as the IP filtering approach used in [8].

We also point out that before carrying out the experiments in this paper, we formally modeled the defense and the Coordinated Call Attack in the computational tool Maude [13] and used statistical model-checking methods [21] to verify it using Monte Carlo simulations. This is reported in another work [15].

There are many directions for future work. We will investigate how one can add additional information, such as the request IP origin, call history of users, into account to improve the precision of our defense. We are also investigating other attacks on VoIP

service that emulate legitimate clients, such as the prank call attack [6]. We are also incorporating other optimizations in SeVen investigating ways to better integrate it with Asterisk. Finally, we are also investigating how SeVen can be incorporated into existing services such as those used by Fone@RNP.

Acknowledgments: The authors were partially funded by the RNP. Lemos, Nigam and Fonseca were partially funded by CAPES and CNPq. Dantas was supported by the Hessian excellence initiative LOEWE at the Center for Advanced Security Research Darmstadt (CASED).

References

[1] Asterisk private branch exchange. http://www.asterisk.org//. Accessed: 2015-27-09.

[2] Cyber threat bulletin: Boston hospital TDoS attack. http://voipsecurityblog.typepad.com/files/cyber-threat-bulletin-13-06-boston-hospital-telephony-denial-of-service-attack.pdf. Accessed: 2015-27-09.

[3] Pindrop: Protecting your call centers against phone fraud social engineering. https://www.pindrop.com/wp-content/uploads/2016/01/pindrop_overview_whitepaper_fi_20141121_v2.pdf. Accessed: 2015-27-09.

[4] Securelogix: Telephony denial of service (tdos) solutions. http://www.securelogix.com/solutions/telephony-denial-of-service-TDoS.html. Accessed: 2015-27-09.

[5] Sipp: Sip traffic generator. http://sipp.sourceforge.net. Accessed: 2015-27-09.

[6] Situational advisory: Recent telephony denial of services (tdos) attacks. http://voipsecurityblog.typepad.com/files/ky-fusion_tdos_3-29-13-2.pdf/. Accessed: 2015-27-09.

[7] TDoS- extortionists jam phone lines of public services including hospitals. https://nakedsecurity.sophos.com/pt/2014/01/22/tdos-extortionists-jam-phone-lines-of-public-services-including-hospitals/. Accessed: 2015-27-09.

[8] Tdos attack mitigation. http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube_proto/configuration/15-mt/cube-proto-15-mt-book/voi-cube-tdos-attack-mitigation.pdf. Accessed: 2015-27-09.

[9] Transnexus nexoss. http://transnexus.com/telephony-denial-service-attacks/. Accessed: 2015-27-09.

[10] *Designing Optimal Voice Networks for Businesses, Government, and Telephone Companies*. 1980.

[11] *The Surging Threat of Telephony Denial of Service Attacks*, (accessed Setember 28, 2015). http://voipsecurityblog.typepad.com/files/tdos_paper_4-11-13.pdf.

[12] Lawrence Brown, Noah Gans, Avishai Mandelbaum, Anat Sakov, Haipeng Shen, Sergey Zeltyn, and Linda Zhao. The title of the work. *Statistical Analysis of a Telephone Call Center: A Queueing Science Perspective*, (100):36–50, 2002.

[13] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer, 2007.

[14] Yuri Gil Dantas. Estratégias para tratamento de ataques de negação de serviço na camada de aplicação em redes ip. Master Thesis in Portuguese, 2014.

[15] Yuri Gil Dantas, Marcilio O. O. Lemos, Iguatemi Fonseca, and Vivek Nigam. Formal specification and verification of a selective defense for tdos attacks. In *11th International Workshop on Rewriting Logic and its Applications (WRLA)*, 2016.

[16] Yuri Gil Dantas, Vivek Nigam, and Iguatemi E. Fonseca. A selective defense for application layer ddos attacks. In *IEEE JISIC 2014*, pages 75–82, 2014.

[17] S. Ehlert, Chengjian Wang, T. Magedanz, and D. Sisalem. Specification-based denial-of-service detection for sip voice-over-ip networks. In *ICIMP '08*, 2008.

[18] Do-Yoon Ha, Hwan-Kuk Kim, Kyoung-Hee Ko, Chang-Yong Lee, Jeong-Wook Kim, and Hyun-Cheol Jeong. Design and implementation of sip-aware ddos attack detection system. In *ICIS '09*, pages 1167–1171, New York, NY, USA, 2009. ACM.

[19] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *CoRR*, abs/1109.3627, 2011.

[20] r-u-dead yet. https://code.google.com/p/r-u-dead-yet/. 2013.

[21] Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 2005.

[22] Ravinder Shankesi, Musab AlTurki, Ralf Sasse, Carl A. Gunter, and José Meseguer. Model-checking DoS amplification for VoIP session initiation. In *ESORICS*, pages 390–405, 2009.

[23] slowloris. http://ha.ckers.org/slowloris/. 2013.

[24] slowread. https://code.google.com/p/slowhttptest/. 2013.

[25] J. Stanek and L. Kencl. Sipp-dd: Sip ddos flood-attack simulation tool. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–7, July 2011.

[26] Jin Tang, Yu Cheng, and Yong Hao. Detection and prevention of SIP flooding attacks in voice over IP networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1161–1169, March 2012.

[27] Jin Tang, Yu Cheng, Yong Hao, and Wei Song. SIP flooding attack detection with a multi-dimensional sketch design. *Dependable and Secure Computing, IEEE Transactions on*, 11(6):582–595, Nov 2014.

[28] Jin Tang, Yu Cheng, and Chi Zhou. Sketch-based sip flooding detection using hellinger distance. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6, Nov 2009.

[29] Xiao-Yu Wan, Zhang Li, and Zi-Fu Fan. A SIP dos flooding attack defense mechanism based on priority class queue. In *WCNIS 2010*, pages 428–431, June 2010.

[30] Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Communications Surveys and Tutorials*, 15(4):2046–2069, 2013.

[31] Fan Zi-Fu, Yang Jun-Rong, and Wan Xiao-Yu. A SIP dos flooding attack defense mechanism based on custom weighted fair queue scheduling. In *ICMT 2010*, pages 1–4, 2010.

Proteção de dados sensíveis através do isolamento de processos arbitrários no kernel Linux

Otávio Augusto Silva[1], **André Grégio**[1,2], **Paulo Lício de Geus**[1]

[1] Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251 – 13083-852 – Campinas – SP – Brasil

[2]Centro de Tecnologia da Informação Renato Archer (CTI/MCTI)
Rod. D. Pedro I (SP-65), KM 143,6 – 13069-901 – Campinas – SP – Brasil

{otavios, gregio, paulo}@lasca.ic.unicamp.br

Abstract. *Linux management policies raise issues such as security decisions being delegated to object owners. Existing frameworks aim to remediate these issues by restricting access to kernel objects (e.g., files) or replacing the kernel code for a more secure version. Although they may prevent a broad set of attacks, they do not mitigate privacy leaks, specially those that abuse the operating system trust base. In this article, we propose a mechanism to ensure that predefined rules are applied on sensitive processes for which the OS trust level is not enough, and the breaking of rules does not raise Mandatory Access Control conflict warnings. Our deployed module hooks the system call table, uses the LSM framework and extends other approaches (AppArmor and Gresecurity).*

Resumo. *As políticas de gerenciamento do Linux podem causar problemas, como decisões de segurança delegadas aos proprietários de um objeto. Mecanismos existentes tentam remedia-los via restrição de acesso a objetos de kernel e substituição deste por um mais seguro. Embora evitem alguns ataques, tais mecanismos não mitigam violações de privacidade, especialmente as que abusam a base de confiança do SO. Neste artigo, propõe-se um mecanismo que aplique regras pré-definidas em processos sensíveis cujo nível de confiança no SO não é suficiente e a quebra das regras não gere alertas de Mandatory Access Control. O módulo proposto intercepta a tabela de chamadas de sistema, usa o framework LSM e estende outras abordagens (AppArmor e Gresecurity).*

1. Introdução

A crescente modernização e monetização dos atacantes traz a necessidade de que todo *software* seja projetado com conceitos de segurança em mente. Assim, um ambiente computacional—doméstico, industrial, governamental ou corporativo—é tão seguro quanto seu *software* mais vulnerável. Como consequência da sofisticação dos ataques, os investimentos em segurança computacional têm crescido ano a ano. Relatórios como [Morgan 2015] preveem um salto no investimento de US$ 77 bilhões em 2015 para cerca de US$170 bilhões em 2020 no mercado norte-americano.

Parte do *software* altamente utilizado possui, em alguma etapa do desenvolvimento, a busca e resolução de falhas de segurança. Além disso, faz-se uso de mecanismos externos (e.g., *security frameworks*) para fornecer um certo nível de integridade e

determinismo durante a execução dos programas de forma a prover um ambiente mais seguro contra ataques. No entanto, mesmo esses ambientes preparados para resistir contra tentativas de violação à segurança podem ser comprometidos por vazamento de informações privilegiadas, brechas na interação baseada na confiança entre os processos do sistema operacional (como compartilhamento de memória) ou problemas em permissões no sistema de arquivos envolvendo processos de um mesmo usuário.

As violações da confidencialidade dos dados relacionados a um processo ocorrem devido as permissões de acesso a um objeto serem partilhadas entre processos de um mesmo grupo, ou entre aqueles que têm o mesmo nível de privilégio, como processos criados pelo mesmo usuário. Esse tipo de violação não é, em geral, causado por *bugs* existentes no sistema operacional, mas como consequência da otimização de recursos presentes nos sistemas operacionais modernos.

O projeto de sistemas operacionais modernos, como o *Linux*, visa muito mais do que fornecer uma interface para que o usuário interaja com o *hardware*. O objetivo é prover uma interface para diversos usuários simultaneamente, com mecanismos e estruturas para o compartilhamento transparente dos recursos limitados do sistema. Essa abstração na concorrência leva diversos processos (de vários usuários) a manterem instâncias de um mesmo recurso, por exemplo um arquivo em disco, em seus contextos de uso. Mediante a grande necessidade de otimização, minimização do uso e alocação de recursos no sistema operacional, tais contextos por diversas vezes são implicitamente compartilhados entre processos com os mesmos privilégios, principalmente aqueles de um mesmo usuário.

O exemplo mais simples de que a confiança atribuída por processos de um mesmo nível de privilégios pode ser abusada e levar ao vazamento de informações é o sistema de arquivos, haja vista que processos de um mesmo usuário podem, no mínimo, acessar os mesmos arquivos e assim abusarem da confiança dada pelo sistema operacional. Entretanto, a segurança provida a um processo pelo *kernel* ou por um *security framework* é passível de ser comprometida pela leitura das entradas pertencentes a processos de um mesmo usuário no *procfs* [Nguyen 2004].

O *procfs* é referido como um pseudo-sistema de arquivos que contém informação sobre processos e sobre o *Linux kernel*. Ele não contém arquivos em armazenamento secundário, mas arquivos virtuais referentes às informações do sistema em tempo de execução (gerenciamento de memória, dispositivos montados, etc.). Ele também contém entradas referentes a cada processo em execução, onde é possível encontrar informações como memória mapeada (incluindo a posição das bibliotecas compartilhadas), arquivos abertos, dispositivos montados pelo processo, entre outras. De posse dessas informações, um processo poderia descobrir quais áreas de memória um outro processo de um mesmo usuário está utilizando, sendo capaz de reduzir, ou mesmo anular, a complexidade de ataques contra a ASLR[1] de um determinado processo.

Neste artigo foram estudados os principais mecanismos de segurança capazes de proteger processos contra falhas de *software* ou de privacidade, seja no nível de usuário ou de *kernel*. Esse estudo revelou a existência de limitações quanto ao isolamento entre processos, onde a violação à privacidade não é tida como ataque pelos mecanismos de segurança avaliados. As principais contribuições do presente trabalho são:

[1] *Address Space Layout Randomization* [Hund et al. 2013].

- Comparar os principais mecanismos de segurança quanto à aplicação, modelo de proteção e limitações.
- Apresentar a proposta e o desenvolvimento de um mecanismo de isolamento e segurança de processos que sane parte das limitações encontradas no estudo comparativo, tendo a privacidade dos dados de um processo como principal requisito.
- Introduzir um estudo de caso do mecanismo proposto em conjunto com o *Grsecurity* para proteção do ambiente em um cenário de ataques não previamente conhecidos pelo sistema.

Este artigo está dividido da seguinte forma: na Seção 2 são apresentados os *security frameworks* existentes na indústria e na academia utilizados para tratar de parte dos problemas de segurança aos quais um processo está sujeito; na Seção 3, são introduzidos o mecanismo proposto, sua arquitetura e aspectos da abordagem usada para aumentar a privacidade dos dados em um ambiente de execução; na Seção 4 são mostrados os testes feitos para comparar algumas abordagens existentes e o mecanismo proposto, bem como os resultados obtidos relacionados a efetividade e sobrecarga; na Seção 5, são feitas as considerações finais.

2. Trabalhos Relacionados

O *kernel Linux* fornece uma *API* para segurança—o *Linux Security Modules Framework*(*LSM* [Morris 2013a])—que permite a monitoração e controle do acesso aos objetos do *kernel* ou dos processos. Esses objetos podem variar de arquivos no disco até *kernel capabilities*, como a capacidade de interagir com *sockets*. Através desse *framework* é possível registrar um módulo no *kernel*, a fim de supervisionar ações em outros subsistemas (e.g., sistema de arquivos) e assim ser capaz de aplicar regras de segurança.

Exemplos de *security frameworks* que utilizam o *LSM* são o *SELinux* [Morris 2013b] e *AppArmor*[AppArmor 2013]. Esses dois sistemas criam, em conjunto com o *LSM*, um controle de acesso por restrição (*MAC*[2]) capaz de limitar o que um processo pode fazer, seja com objetos em nível de usuário ou do *kernel*. Com isso, incrementa-se o gerenciamento do acesso e as permissões já existentes no Linux, ao atribuir aos processos *labels* ou *tags* relacionadas às permissões concedidas.

O *SELinux* segue o modelo do menor privilégio. Em um cenário de práticas rigorosas, tudo é negado por padrão e, somente após a escrita de uma série de exceções nas políticas, libera-se o estritamente necessário para um processo funcionar. Tais políticas são escritas especificamente para cada executável cujos processos deseja-se controlar. Os executáveis são identificados pelos seus respectivos *inodes*, e as políticas aplicadas em dois regimes, *Enforcing* e *Permissive*. No modo *Enforcing*, o *SELinux* impede toda ação que infringe uma política de segurança e a reporta como um evento a ser logado; já no modo *Permissive*, os eventos são apenas reportados. O conjunto de regras e políticas possíveis é muito vasto, sendo considerado extremamente complexo quando comparado a outros sistemas que fazem uso do *LSM* e cumprem o mesmo objetivo, porém com regras mais simples e diretas.

O *AppArmor*, por sua vez, também faz uso do *LSM*, mas possui regras mais simples. Apesar de não ser o único, é o mais utilizado deles. Outros *security frameworks* que

[2]Mandatory Access Control

foram integrados ao *kernel vanilla* são o Smack [Schaufler 2011], que propõe um sistema de MAC simplista para ambientes embarcados com poucos recursos de *hardware*, e o Tomoyo [Corporation 2015], que possui uma aplicação balanceada tanto em auditoria e métricas do sistema, quanto em segurança. O *AppArmor* possui dois modos de operação, *Complain* e *Enforce*, onde o primeiro apenas registra violações às regras, enquanto o outro rejeita o acesso aos objetos controlados. Diferente do *SELinux*, o *AppArmor* utiliza *profiles* para gerenciar as regras de cada processo. Esses *profiles* são válidos para quaisquer processos cujo executável corresponda a um caminho específico. É nesses *profiles* que estão contidas todas as políticas de acesso, objetos controlados e mecanismo de acesso disponível para o executável em questão.

Apesar das diferenças na configuração, aplicação dos *labels* ou identificação dos objetos e processos protegidos, os *frameworks SELinux* e *AppArmor*, como usam o *LSM*, são equivalentes em termos de segurança fornecida. [Chen et al. 2009] demonstram a equivalência na segurança ao comparar os mecanismos utilizados por ambos os sistemas, porém com variações no *overhead* resultante de cada sistema. [SUSE 2016] apresenta uma tabela comparativa das principais *features* de ambos os sistemas.

O *Grsecurity* [Spengler 2015c] é um sistema de segurança cuja abordagem difere da usada por sistemas baseados em *LSM*, pois faz seus próprios *hooks* e pontos de entrada no *kernel* e, apesar de também fornecer um sistema equivalente ao *MAC*, sua proteção vai além: como o *Grsecurity* possui um conjunto de *patches* do *kernel Linux*, ele faz mudanças de segurança no código do *kernel* em quase todos subsistemas, aumentando a proteção nos níveis de usuário e de *kernel*. Com essa abordagem, o *Grsecurity* protege todo o sistema contra *exploits*, ataques inerentes ao funcionamento do *kernel Linux* (e.g.,jitspray [McAllister 2012]) e manipulação dos dados de um processo por outro. Os principais tipos de proteção alcançados por esse sistema encontram-se descritas em [Spengler 2015b], enquanto que [Fox et al. 2009a] comparam a proteção provida entre *Grsecurity* e *SELinux*.

Para determinar o comportamento esperado de um processo e a decisão a ser tomada em caso de anomalia, o *Grsecurity* aplica aprendizado de máquina com regras baseadas em *RBAC* (*Role-Based Access Control*). O *PaX*, responsável pela gerência de segurança do processos em nível de usuário, possui dois modos de operação: aprendizado e produção. No modo aprendizado, o sistema é mantido fora de um ambiente "não-confiável" (produção), no qual todas as operações realizadas pelos processos são ditas confiáveis e farão parte da base de conhecimento.

A matriz de comparação presente em [Spengler 2015a] torna possível visualizar as diferenças entre *Grsecurity*, *SELinux* e *AppArmor*. Por exemplo, *Grsecurity* pode proteger um processo tal como qualquer sistema baseado em *LSM*, mas como protege o *kernel*, acaba por proteger também o subsistema do próprio *LSM*.

[Onarlioglu et al. 2013] propõem um sistema—*PrivExec*—que utiliza-se de mecanismos como restrições por IPC (*Inter Process Communication*) e *eCryptfs* [Kirkland 2012] para proteger um processo e seus dados. O *PrivExec* divide os processos entre os protegidos e os outros, onde a diferenciação ocorre na existência de uma *flag* adicional na *task_struct* e em uma chave simétrica, utilizada no *eCryptfs* naqueles que são protegidos. Esse sistema fornece o isolamento dos dados persistentes

através da separação da visão do *root filesystem*, com uso do *OverlayFS* [Brown 2015], e de uma versão alterada do *eCryptfs* para criptografar os arquivos do processo. Essa abordagem estende-se também ás páginas de *swap* de um processo, porém o *PrivExec* não protege diretamente nenhuma outra página de memória, com exclusão àquelas utilizadas em IPC. O isolamento por *IPC* ocorre através de restrições aos grupos de processos que podem iniciar essa comunicação. Um processo pertence a um grupo caso ele possua a chave simétrica compartilhada no grupo. Essa chave é compartilhada pelo primeiro processo a integrar o grupo, o qual usualmente é o processo a ser protegido, e cujas dependências foram incluídas neste grupo. A política implementada permite que processos do mesmo grupo se comuniquem livremente, enquanto outros processos recebem um erro de permissão, na leitura ou na escrita, quando feitas em outros grupos privados. As estruturas do *kernel Linux* que foram modificadas para respeitar essa política foram: *UNIX SysV & POSIX shared memory, message queues* e *UNIX domain sockets*.

O *PrivExec* consegue o isolamento de um processo arbitrário, garantindo o sigilo dos arquivos manipulados tanto no disco quanto no *swap* e da comunicação via *IPC*. Entretanto, essa implementação apenas provê segurança nos protocolos de *IPC* mais comuns, assim como não garante qualquer isolamento contra leitura ou vazamento de memória perante outros processos. As limitações do *PrivExec* incluem o fato de ser uma proposta conceitual na qual não se considera resistência em caso de ataques, falhas do sistema ou presença de *malware* previamente à sua instalação.

2.1. Limitações dos trabalhos relacionados

Os *security frameworks* apresentados representam o estado da arte na proteção de processos, seja contra ataques por meio de falhas de *software*, ou contra ataques que mudam o comportamento esperado do *software* ao longo de sua execução. Embora tais *frameworks* possam ser bem-sucedidos na prevenção da exploração de muitas vulnerabilidades, existem classes inteiras delas que não podem ser evitadas, como *kernel capabilities* desnecessárias a um processo ou *race conditions*.

Isto ocorre devido parcialmente à limitação inata de todos os sistemas de controle de acesso, como descrito por [Ausanka-Crues 2006]: eles precisam ser tão automáticos quanto possível e possuir configurações legíveis e facilmente editáveis, além de um rigoroso e estrito controle de objetos, de modo a fazer cumprir as políticas de segurança ao eliminar erros do administrador. Desta forma, ocasionalmente uma vulnerabilidade precisa ser tratada pela elaboração de novas regras, em geral porque novos objetos ou *capabilities* tornaram-se alvo dessa vulnerabilidade e as antigas regras não foram suficientes. Essa limitação está presente no *Grsecurity*, conforme mostrado por [Bugliesi et al. 2012] ao fazerem uma profunda análise, incluindo limitações e *trade-off*, da gerência por *RBAC* em relação a *MAC*. Em [Fox et al. 2009b], corrobora-se tais limitações ao se contrastar a implementação do *SELinux* e do *Grsecurity* quanto à gerência de acesso.

Cabe ressaltar que os *frameworks* acima mencionados não tratam dos recursos dos processos controlados por eles em relação a outros processos do sistema, caso esses últimos também não sejam controlados. Ambos, *LSM* e *Grsecurity*, controlam o que um processo controlado pode acessar, porém sem interferir no que um processo com os mesmos privilégios do processo controlado pode acessar, caso esse acesso não viole nenhuma política de segurança. Esse modo de projeto é um problema mais de privacidade

do que de segurança: caso um processo controlado seja subvertido, ele não será capaz de fazer mais do que ele é permitido pelas políticas de segurança. Porém, é possível um outro processo obter dados daquele processo controlado sem de fato influenciar em sua execução, isto é, sem causar correspondência com nenhuma regra de segurança, dado que não se trata de um cenário identificado como ataque pelos *frameworks*.

Um bom exemplo é o servidor Web *Apache*, o qual possui um único arquivo de configuração de políticas nos *frameworks* supracitados, mas geralmente é executado como mais de uma instância. Cada instância cuida geralmente de vários *virtual hosts* e é um processo de um mesmo usuário. Neste cenário, uma instância pode interferir com dados de outras instâncias, pois para os sistemas de segurança, todos as instâncias tem o mesmo nível de acesso e privilégios. Isso acontece devido à forma como cada processo é controlado com base no executável que deu-lhe origem, através do *path* ou *inode* no sistema de arquivos. Logo, uma instância do *Apache* pode sobrescrever arquivos de outra instância, tais como *logs*, *Unix socket lockfiles*, configurações, ou mesmo nos arquivos hospedados.

Embora o *Grsecurity*, diferente do *LSM*, não possua apenas um gerente de acesso para aplicar suas políticas e possa tratar cada processo como uma instância diferente no sistema operacional, ele não irá negar o acesso de uma instância à outra. Isso acontece porque o *Grsecurity* não pode coincidir esse comportamento com um perfil de ataque, fazendo com que apenas verifique os direitos de acesso dos recursos que o processo está autorizado a utilizar. Assim, ele também não consegue garantir a privacidade dos dados de um processo quando o acesso a estes é inerentemente permitido para outro processo. Este cenário sempre acontece quando o projeto do sistema operacional espera uma base de confiança na relação entre processos, tais como a relação pai-filho ou processos do mesmo usuário.

O mecanismo proposto neste artigo sanou algumas das limitações citadas ao retirar a proteção por meio de gerências de acesso e por perfil de ataques, passando-a para a gerência da posse dos dados. Por exemplo, de acordo com a gerência de permissões de um objeto, como um arquivo ou um *socket*, este só pode ser acessado caso o processo consiga decifra-lo, e não de acordo com seus privilégios no sistema operacional.

3. Arquitetura do Mecanismo Proposto

Neste artigo, propõe-se um mecanismo de isolamento de processos arbitrários a partir do contexto de outros processos, para assim aumentar a segurança e privacidade mesmo no caso de processos que compartilham os mesmos privilégios. A proposta é complementar ao tipo de proteção provida por outros *frameworks*, como o *Grsecurity*, pois resolve algumas de suas limitações, principalmente quando a base de confiança do sistema operacional não é suficiente, como mostrado anteriormente.

O mecanismo de segurança aqui proposto, chamado a partir de agora de OSPI (*Operating system Security Process Isolator*), foi desenvolvido como um módulo do *kernel Linux* para arquitetura x86 com suporte à extensão *AES-NI* e um gestor de nível de usuário. Sua arquitetura permite que os processos utilizem características especiais de segurança introduzidas no *kernel* pelo módulo, o qual tem por objetivo segregar os processos em grupos, de modo a aplicar políticas especiais de segurança—cada objeto manipulado por um processo, em nível de usuário ou não, é exclusivamente de sua propriedade ou compartilhado entre um grupo predefinido de processos. Para isso, o OSPI faz uso prin-

cipalmente dos seguintes recursos do *kernel*: *syscall hooking*, *LSM framework*, *CGroups (control groups)* e *CryptoAPI*. Além das estruturas do *kernel* citadas, o OSPI também inclui as seguintes estruturas lógicas: uma lista completa de processos controlados, uma lista global de estruturas do *kernel* controladas, uma lista de símbolos do *kernel* por processo e máscaras de bits por processo que mapeiam as permissões e políticas de cada processo.

A utilização das referidas estruturas permite que o módulo controle o acesso e as permissões de cada processo ou grupo de processos cujas permissões são compartilhadas. Esse controle é realizado nas estruturas cujo comportamento padrão deva ser alterado seguindo políticas de segurança, como a *syscall* `process_vm_readv`, que deve retornar sucesso, porém não deve retornar um bloco de memória quando destinada a um processo protegido, independentemente das permissões do usuário. As técnicas para interceptar e alterar as estruturas do *kernel* incluem o *hooking* ou acesso direto aos subsistemas do *kernel*, como no *virtual filesystem*, impedindo o mapeamento de um *incore inode* por um outro módulo do *kernel*, através da retirada de sua referência na lista de *incore inodes*.

Ao contrário de outros mecanismos de segurança do Linux, o controle de recursos proposto não utiliza os locais dos objetos (e.g., caminho no sistema de arquivos), mas irá utilizar as informações de posse presentes nas estruturas do módulo. Para conseguir isso, o módulo decidirá se, e como, um recurso será compartilhado, de acordo com a máscara de acesso do primeiro processo que obtiver a posse do objeto. Essa posse é atribuída através da relação entre processo controlado e objeto a ser manipulado, dado que tal objeto não foi manipulado por nenhum outro processo ou grupo de processos até então.

A abordagem de controle é feita no acesso da camada de *syscall*, fornecendo uma interface simples para conceder o controle sobre cada processo no sistema, pois permite mudar como um conjunto de *syscalls* se comportam, de acordo com o processo que as executarem. Com isso em mente, se um processo tentar manipular qualquer objeto no espaço de usuário, seu pedido terá de passar primeiro pelo OSPI, pois os *hooks* são colocados em *syscalls* como *open*, *read*,*write*, *exec*, *mmap*, etc. Com esses *hooks*, o módulo pode verificar se quem faz a chamada é um processo controlado, ou se ele está manipulando recursos controlados, aplicando restrições se necessário, e, em seguida, passar a execução para o código original da *syscall*. Deste ponto em diante, outras estruturas de segurança, como o *Grsecurity*, podem aplicar as suas políticas de segurança, de modo que o OSPI irá fortalecer um subsistema de segurança.

Dentro do *hook* de uma *syscall*, o módulo pode atuar de duas maneiras distintas: *managed* ou *restricted*. No modo *managed*, as políticas são aplicadas de acordo com um sistema simples de MAC ao conceder ou negar acesso aos objetos controlados, retornando um erro na *syscall* quando negado. Neste modo também é possível o uso de um objeto efêmero, criado exclusivamente como um objeto virtual. Seguindo a máscara de permissões, um processo pode ter todas as suas gravações confinados ao sistema de arquivos virtual, em se tratando de arquivos ou *buffers* geridos pelo *kernel*. Esta execução efêmera permite que um processo sustente uma execução privada de forma transparente, de modo que todas as alterações são confinadas na memória do *kernel* até o fim da execução, com o processo alheio a tal controle.

O modo *restricted* garante a privacidade e segurança de uma forma mais intru-

siva: ele criptografa os dados ao escrever e decifra ao ler, de todos os objetos tocados por um determinado processo. Isso impede que outros processos, independentemente de sua permissão, consigam ler dados em texto claro de um processo executado no modo restrito. Este modo pode ser persistente, de maneira que o processo ainda possa ler os dados manipulados mesmo se o sistema for reiniciado. Caso a persistência dos dados faça-se necessária, os diferentes processos lançados pelo mesmo executável terão a mesma chave de criptografia, invalidando assim o requisito que garante o isolamento de dados mesmo para diferentes instâncias do mesmo executável.

A cifragem e decifragem dos dados ocorrem em *syscalls* como `read` e `write`, ou `msgget` e `msgsnd`(IPC), usando a *CriptoAPI* [James Morris 2015] do *kernel Linux*. Essa *API* suporta diversos criptossistemas e funções de *hashing*, utilizando sempre as implementações e abordagens dos algoritmos que melhor aproveitem os dispositivos de *hardware* presentes. As operações sobre os objetos controlados ocorrem utilizando uma chave no criptossitema *AES128* disponibilizado pelo *kernel*. Uma vez que tais operações são feitas internamente nas *syscalls* controladas, essas são transparentes para os processos envolvidos. A chave utilizada é composta de duas outras chaves, a chave do módulo, que nunca é armazenada na memória principal, e a chave privada, única para cada processo controlado.

A geração da chave privada do processo pode ocorrer de duas formas, através do gerente em nível de usuário que devolve o *hash* MD5 do *ELF* relativo ao executável do processo, ou através do módulo utilizando informações da *task_struct* do processo. Devido a esse módulo ainda ser um trabalho em andamento, a gerência em nível de usuário funciona para quaisquer *kernels* da família *3* e *4*, enquanto a gerência exclusiva via módulo funciona apenas na família *3* do *kernel Linux*. O módulo do OSPI utiliza um tipo `mm_segment_t` da estrutura `thread_info` e um tipo `mm_struct` da estrutura `task_struct` para obter a seção *.text* do processo e assim calcular o seu *hash* MD5. Um detalhe importante referente a implementação é que o endereço apontado pela varável `mm_segment_t` deverá ser convertido para o endereço real, dado que a seção *.text* começaria no endereço virtual *0x00000000*. Essa conversão é feita pela tupla de funções do *kernel* `get_fs` e `set_fs`.

Independente da abordagem utilizada, o *hash* é capaz de representar todo o código disponível a ser executado pela *task* e assim, com alto grau de confiança, ser a chave única de um processo controlado mesmo quando a máquina reinicie e todos os identificadores do processo mudem. Caso a execução não tenha o bit de persistência, uma semente pseudo-aleatória provida pela entropia do *kernel* é adicionada como *salt* ao *hash*, tornando possível que processos de um mesmo executável mantenham instâncias privadas entre si. Uma vez que uma execução não persistente termine, todos os objetos manipulados pelo processo em questão podem ser considerados perdidos, em vista da criptografia utilizada e da perda da chave.

A chave privada do processo não pode ser usada diretamente como a chave no criptossistema *AES128*, pois o espaço de bits suscetíveis a colisões inerente ao *MD5* enfraqueceria a confidencialidade do *AES128*. Desta forma, a chave utilizada é o resultado da operação XOR (ou-exclusivo) das chaves privadas do processo e do módulo proposto. O uso de tal operação, para a geração de chaves únicas a partir de uma chave *tamper-proof* e outra sem tal garantia, não enfraquece ou expõe a primeira chave, ao passo que torna

as chaves geradas diferentes tanto quanto a segunda chave da operação. [Steel 2005] demonstra como podem ocorrer ataques a partir da extração de características das chaves geradas por meio de algoritmos que usam operações *XOR* em *Security APIs* largamente utilizadas na infraestrutura bancária. Portanto, a segurança contra a dedução da chave gerada no presente trabalho depende da segurança do algoritmo utilizado para o cálculo do *hash* e do algoritmo que gerou a chave do módulo. Tal segurança pode ser violada se não houver preservação mútua das características presentes nas saídas dos algoritmos, ou se as características da entrada não forem preservadas na saída.

O estudo supracitado apresenta outra fragilidade adicional à extração de características: a propriedade do inverso, na qual o resultado da operação será sempre "0" (zero). Esse ataque depende do conhecimento da chave resultante e de uma das chaves da composição, no caso a chave privada de um processo, em vista da facilidade de calcula-la. Porém, a chave do processo sozinha não tem a intenção de ser sigilosa, e de fato pode ser conhecida, dado que a chave resultante e a chave do módulo nunca residem a memória principal. Logo, ambas as chaves só podem ser conhecidas em nível de *kernel* por meio da leitura de registradores. A chave resultante é produto de instruções do processador e, portanto, é armazenada em registradores. Essa chave imediatamente após ser gerada serve como parâmetro no *AES128* através de instruções AES-NI (*Advanced Encryption Standard New Instructions*) presentes no processador. Isso faz com que ela não seja armazenada na memória principal.

A chave do OSPI é utilizada para todos os processos, mas não deve ser gerada pelo módulo e sim por uma boa fonte de entropia. Pretende-se, em uma versão posterior, que o módulo seja capaz de ler a chave nas primeiras fases do *boot*—por meio de um dispositivo de armazenamento de confiança ou um *hardware* gerador—e, após leitura preencha com "0" cada *buffer* usado para transporte da chave, armazenando-a, por fim, nos registradores de *debug* do processador (`DR0` a `DR3`). Esses registradores não podem ser utilizados fora do modo privilegiado do processador, a não ser que um *debugger* executando como *root* os acesse via capacidades especiais providas pelo *kernel Linux*.

O OSPI usa diretamente as funções do *kernel* para desabilitar e impedir o uso dos registradores de *debug* por meio das seguintes funções: `arch_uninstall_hw_breakpoint`; `hw_breakpoint_restore`; `arch_install_hw_breakpoint` (irá sofrer *hook* e chamar `hw_breakpoint_restore`). Desta forma, inviabiliza-se o uso de *hardware breakpoints* ao mesmo tempo em que se aumenta o nível de segurança do sistema, pois há trabalhos que mostram como *rootkits* abusam dessa funcionalidade de forma a persistir no sistema alvo e comprometer tanto o OSPI quanto o *Grsecurity* [Emil Persson 2012], [Phrack.0x0c41 2008].

A Figura 1 ilustra as interações entre os diversos níveis de processos presente em um sistema. As interações representadas pelas arestas pretas pertencem a processos não-controlados, enquanto que as em escala de cinza representam os processos sujeitos a alguma forma de controle. Arestas cinza-claro ou cinza-escura são interações com objetos pertencentes aos grupos "0" ou "1", cujos dados são cifrados ou decifrados em decorrência do parâmetro presente em uma *syscall* controlada. As interações negadas são representadas por arestas esbranquiçadas, nas quais os processos não possuem entrada na lista de processos controlados e, portanto, não possuem uma chave simétrica.

Caso fossem processos do *grupo0* tentando acessar objetos do *grupo1*, mesmo sem uma regra específica no *MAC* para coibir esse tipo de atividade, o objeto acessado em questão seria decifrado pela chave errada, não revelando assim suas informações (o resultado de tal decifragem seria ininteligível). As arestas cinza-médio indicam uma interação anômala, enquanto que a aresta tracejada indica um ataque não-detectado pelo *Grsecurity*.

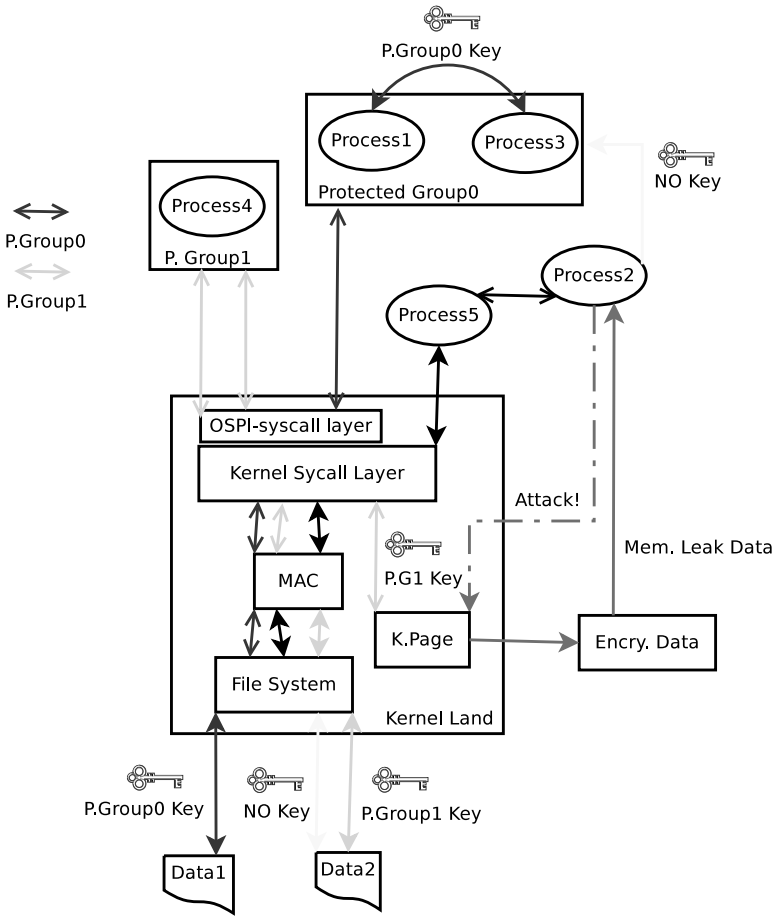


Figura 1. Diagrama de interações entre grupos de processos controlados e suas chaves com processos fora do escopo de controle do OSPI.

4. Testes e Resultados

O desenvolvimento do módulo está em andamento, almejando um mecanismo com fina granularidade e proteção em baixo nível de forma a impor as necessidades de segurança de processos sensíveis ou especiais. O mecanismo foi projetado para ser capaz de aumentar o nível de privacidade dos dados nos casos onde os mecanismos de proteção existentes falham. Além disso, a combinação da abordagem proposta neste artigo com o *Grsecurity* faz com que um determinado processo possa ser protegido contra violações de privacidade e ataques em cenários incomuns.

Para testar o protótipo desenvolvido em sua versão mais atual, bem como a atuação de outros mecanismos de segurança disponíveis para instalação, utilizou-se um cenário de exploração de virtualização com *Docker*: *docker container breakout* [Turnbull 2014].

Uma vez que o ataque em questão não possui uma regra de detecção correspondente a ataque conhecido até o momento da escrita deste artigo, tanto o *Grsecurity* quanto o *AppArmor* foram incapazes de evitar que este fosse completado com sucesso. O *profile* do *AppArmor*, até a versão do *Docker 0.11*, permitia bloquear *kernel capabilities* através de sua inserção em uma *blacklist*. Porém, a falha contou com uma combinação de *syscalls* e *capabilities* que não estavam na *blacklist* até que o ataque ficou conhecido. Esta combinação causou uma brecha que permitiu a um *docker* executado pelo *root* ser capaz de ler arquivos e diretórios localizados fora do isolamento.

O motivo do sucesso deste ataque em sistemas com *Grsecuriy* e *AppArmor* deu-se pelo fato de que, mesmo que o processo isolado não seja executado como *root* internamente ao *docker*, um processo com privilégio de *root* executa *syscalls* como tal. Isto ocorre devido a virtualização em espaço de usuário compartilhar a mesma *syscall layer* do resto do sistema. Para barrar esse ataque, *AppArmor* recebeu um novo *profile*, em que as *capabilities* foram bloqueadas por padrão (exceto para aqueles em uma *whitelist*).

O OSPI foi capaz de bloquear o ataque sem sequer conhecê-lo. Portanto, o módulo proposto já é capaz de atenuar uma classe inteira de ataques semelhantes, devido à utilização do modo restrito no processo protegido. Assim, um processo que quebre o isolamento provido pelo *Grsecurity* não será capaz de ler qualquer objeto que não seja seu, de acordo com a política de acesso do *OSPI*. Mesmo que o processo consiga subverter as restrições do *MAC* e ter acesso ao objeto, dado que o conteúdo do objeto está cifrado, esse acesso não será bem sucedido no contexto de um ataque de violação da confidencialidade. O cenário da escrita funciona de forma semelhante: como o processo que possui a posse do objeto sempre o recebe na leitura após o modulo decifra-lo, a escrita direta deste objeto por um ataque resultará em dados ininteligíveis para o processo protegido. Desta forma, o processo subvertido não poderá escrever o que pretende, pois mesmo que haja alteração dos dados, esta corrompe o arquivo, mitigando o ataque (por exemplo, a escrita de credenciais em */etc/shadow* resultaria em uma linha inválida e, portanto, inútil para um *login* do atacante).

Embora o *OSPI* ainda não tenha sido amplamente testado, o sucesso alcançado no bloqueio de alguns ataques que não são tratados por outros *frameworks* sem configurações específicas mostra que a abordagem é promissora. Os cenários avaliados—*rootkits* e *docker leak*—valida sua eficácia como um mecanismo complementar para reforçar a segurança de um sistema operacional e seus processos cuja proteção depende dos *frameworks* tradicionais.

Ao longo do projeto, foram feitas aferições acerca do *overhead* resultante dos mecanismos utilizados pelo módulo, a fim de se escolher a abordagem mais eficiente que, ao mesmo tempo, não implicaria em redução na segurança do sistema. As abordagens implementadas, tais como utilizar a fila de processos e objetos estruturadas em uma tabela *hash* e realizar a conversão dos endereços virtuais para os reais na *OSPI syscall layer* previamente à chamada original do *kernel*, reduziram muito o *overhead* e o *turnaround time*[3]. das chamadas de sistema efetuadas. A Tabela 1 ilustra algumas *syscalls* executadas no mesmo *kernel*, mas com *frameworks* de segurança diferentes. O *turnaround time* de uma *syscall* é representado em milissegundos, enquanto que o *overhead* em porcentagem

[3]Tempo necessário para completar um processo, do lançamento para execução até o retorno de sua saída

da chamada em relação ao *kernel* sem nenhuma proteção. Foi utilizado o `SystemTap` e uma adaptação do *script iotime.stp* [RedHat 2007] para obtenção do tempo gasto nas chamadas.

Foram feitas 1024 chamadas de cada uma das *syscalls* representadas na Tabela, utilizando-se *1KB* de dados por meio da ferramenta *syscall fuzzer Trinity*[Jones 2015] quando necessário. A chamada `null I/O` corresponde a execuções das *syscalls* `read` e `write`, porém com o *buffer* vazio. Todas as chamadas realizadas foram feitas de modo pseudo-aleatório, sempre seguindo a semântica dos parâmetros necessários para cada *syscall*. Com isso, elas representam tanto o comportamento esperado quanto o inesperado, simulando uma anomalia que possivelmente é tratada pelos *frameworks* de segurança testados.

Tabela 1. Tabela comparativa com o *turnaround time* em milissegundos das *syscalls* em um sistema protegido pelos *frameworks* OSPI, Grsecurity ou AppArmor em comparação à chamada sem proteção (base). As medidas de*overhead* correspondem à porcentagem do *turnaround time* da chamada protegida em relação à chamada desprotegida.

Syscall	Base	OSPI	Overhead	Grsecurity	Overhead	Apparmor	Overhead
`null I/O`	0,13	0,13	1%	0,13	5%	0,13	12%
`open/close`	2,05	2,05	4%	2,09	2%	4,28	109%
`fork`	131,17	131,43	2%	130,23	1%	134,71	27%
`execve`	386,52	575,91	49%	552,72	43%	587,51	52%
`pipe`	6,67	7,33	10%	7,67	15%	7,53	13%
`read`	8,38	11,98	43%	9,30	11%	9,14	9%
`write`	14,09	22,12	57%	15,49	10%	15,27	8%

5. Conclusão

Neste artigo, discutiu-se os *frameworks* mais utilizados para a proteção da privacidade dos dados de processos em execução em sistemas operacionais Linux, como *Grsecurity* e *AppArmor*. Foram levantadas as limitações de tais *frameworks* em face de alguns ataques que exploram vulnerabilidades de programas causando a exposição de informações sensíveis, seja por vazamento de dados ou pela presença de *malware* em nível privilegiado do sistema operacional. Com a finalidade de aumentar o nível de segurança do usuário, impedindo certas violações à confidencialidade, foi proposto o OSPI, um mecanismo de segurança que isola processos arbitrários do sistema, protegendo assim o contexto de seus dados. Apresentou-se a arquitetura do mecanismo proposto, bem como testes preliminares e seus resultados. Foi possível notar que o mecanismo proposto alcançou, em média, um baixo *overhead* quando comparado com o *AppArmor*, mas obteve taxa de *overhead* superior ao *Grsecurity*. Entretanto, o último resultado era esperado devido ao uso do critpossistema *AES128*, que por si só demanda de um esforço computacional superior ao demandado pelo *Grsecurity*. Porém, a diferença obtida em relação ao *AppArmor*, que possui em média um *overhead* inferior ao *SELinux*, foi notável, principalmente ao se considerar que as chamadas *open/read* apresentaram mais do que o dobro de *turnaround time*. Tal diferença é causada principalmente pelo tempo para se fazer o *parsing* dos arquivos de políticas, inexistente no *OSPI* ou *Grsecurity*. Como um protótipo, os resultados foram promissores e indicam o potencial da continuidade do projeto como um mecanismo viável de proteção complementar.

Referências

[AppArmor 2013] AppArmor, W. (2013). Apparmor project wiki. http://wiki.apparmor.net. Acessado em: 30 Mar. 2016.

[Ausanka-Crues 2006] Ausanka-Crues, R. (2006). Methods for access control: Advances and limitations. Technical report, Harvey Mudd College.

[Brown 2015] Brown, N. (2015). Overlayfs documentation. https://www.kernel.org/doc/Documentation/filesystems/overlayfs.txt. Acessado em: 03 Abr. 2016.

[Bugliesi et al. 2012] Bugliesi, M., Calzavara, S., Focardi, R., and Squarcina, M. (2012). Gran: Model checking grsecurity rbac policies. In Chong, S., editor, *CSF*, pages 126–138. IEEE.

[Chen et al. 2009] Chen, H., Li, N., and Mao, Z. (2009). Analyzing and comparing the protection quality of security enhanced operating systems. In *NDSS*. The Internet Society.

[Corporation 2015] Corporation, N. D. (2015). About tomoyo linux. http://tomoyo.osdn.jp/about.html.en. Acessado em: 30 Mar. 2016.

[Emil Persson 2012] Emil Persson, J. M. (2012). Debug register rootkits. a study of malicious use of the ia-32 debug registers. Technical report, School of Computing,Blekinge Institute of Technology.

[Fox et al. 2009a] Fox, M., Giordano, J., Stotler, L., and Thomas, A. (2009a). Selinux and grsecurity: A case study comparing linux security kernel enhancements.

[Fox et al. 2009b] Fox, M., Giordano, J., Stotler, L., and Thomas, A. (2009b). Selinux and grsecurity: Mandatory access control and access control list implementations.

[Hund et al. 2013] Hund, R., Willems, C., and Holz, T. (2013). Practical timing side channel attacks against kernel space aslr. In *IEEE Symposium on Security and Privacy*, pages 191–205. IEEE Computer Society.

[James Morris 2015] James Morris, David S. Miller, H. X. (2015). Linux kernel cryptographic api. http://lxr.free-electrons.com/source/Documentation/crypto/api-intro.txt. Acessado em: 30 Mar. 2016.

[Jones 2015] Jones, D. (2015). Trinity: Linux system call fuzzer. https://github.com/kernelslacker/trinity. Acessado em: 30 Mar. 2016.

[Kirkland 2012] Kirkland, D. (2012). Ecryptfs documentation. http://ecryptfs.org/documentation.html. Acessado em: 03 Abr. 2016.

[McAllister 2012] McAllister, K. (2012). Attacking hardened linux systems with kernel jit spraying. https://lwn.net/Articles/525609. Acessado em: 30 Mar. 2016.

[Morgan 2015] Morgan, S. C. (2015). Cybersecurity market report q4 2015. http://cybersecurityventures.com/cybersecurity-market-report.

[Morris 2013a] Morris, J. (2013a). Linux security module framework. https://www.linux.com/learn/overview-linux-kernel-security-features. Acessado em: 30 Mar. 2016.

[Morris 2013b] Morris, J. (2013b). Selinux project wiki. http://selinuxproject.org. Acessado em: 30 Mar. 2016.

[Nguyen 2004] Nguyen, B. (2004). Linux proc file system. http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html.

[Onarlioglu et al. 2013] Onarlioglu, K., Mulliner, C., Robertson, W., and Kirda, E. (2013). PrivExec: Private Execution as an Operating System Service. In *IEEE Symposium on Security and Privacy*, San Francisco, CA USA.

[Phrack.0x0c41 2008] Phrack.0x0c41 (2008). Mistifying the debugger ultimate stealthness. http://phrack.org/issues/65/8.html. Acessado em: 30 Mar. 2016.

[RedHat 2007] RedHat (2007). Systemtap script iotime. https://sourceware.org/systemtap/examples/io/iotime.stp. Acessado em: 30 Mar. 2016.

[Schaufler 2011] Schaufler, C. (2011). Smack project description. http://schaufler-ca.com/description_from_the_linux_source_tree. Acessado em: 30 Mar. 2016.

[Spengler 2015a] Spengler, B. (2015a). Grsecurity comparison matrix. https://grsecurity.net/compare.php. Acessado em: 30 Mar. 2016.

[Spengler 2015b] Spengler, B. (2015b). Grsecurity features. https://grsecurity.net/features.php. Acessado em: 30 Mar. 2016.

[Spengler 2015c] Spengler, B. (2015c). Whats is grsecurity. https://grsecurity.net/about. Acessado em: 30 Mar. 2016.

[Steel 2005] Steel, G. (2005). Deduction with XOR constraints in security API modelling. In Nieuwenhuis, R., editor, *Proceedings of the 20th International Conference on Automated Deduction (CADE'05)*, volume 3632 of *Lecture Notes in Artificial Intelligence*, pages 322–336, Tallinn, Estonia. Springer.

[SUSE 2016] SUSE (2016). Apparmor and selinux comparison. https://www.suse.com/support/security/apparmor/features/selinux_comparison.html. Acesso em 11 de abril de 2016.

[Turnbull 2014] Turnbull, J. (2014). Docker Container Breakout Proof-of-Concept Exploit. https://blog.docker.com/2014/06/docker-container-breakout-proof-of-concept-exploit/.

Trilha Principal do SBRC 2016
Sessão Técnica 4
Redes Sem-fio

Reduzindo a Probabilidade de Colisões através do Encurtamento dos Slots em Redes 802.11

Rafael Araújo da Silva, Aldri Luiz dos Santos, Michele Nogueira

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
{rasilva,aldri,michele}@inf.ufpr.br

Abstract. *One of the main challenges on the next generation local wireless networks lies in providing scalable media access control protocols to support the expected high density. Research results have indicated that the current backoff mechanism is ineffective for dense networks, even more under the expected charge growth for the next five years. This paper analyzes the proposal of enlargement on the contention window by reducing the slot time to be implemented in the the 802.11ax amendment. Its goal lies in improving the performance of next generation wireless local area networks over high dense scenarios. Our analysis considers different classes of traffic reflecting on the access categories for data prioritization and their respective windows. Simulation results indicate that the proposal mitigates collision issues inherent to the backoff procedure, and it decreases the frame loss. Our study shows the improvements achieved by the use of this proposal as increased throughput and reduced retransmissions and also alerts to the impact on the clear channel assessment process. Our results contributes with the literature assisting in the advance of the next generation local wireless networks.*

Resumo. *Um dos principais desafios das redes locais sem fio da próxima geração é prover protocolos de acesso ao meio escaláveis para suportar a grande densidade estimada. Resultados de pesquisa indicam que o mecanismo de backoff atual é ineficaz em ambientes de redes densas, ainda mais sob o esperado crescimento nesta carga para os próximos cinco anos. Este trabalho analisa a proposta de alargamento da janela de contenção através da redução da duração dos slots a ser implementada pela emenda 802.11ax. Nossa análise considera as diferentes classes de tráfego, refletindo nas categorias de acesso para priorização de dados e suas respectivas janelas. Os resultados de simulação indicam que a proposta atenua o problema das colisões inerentes ao backoff e diminui a probabilidade de descartes de quadros. Nosso estudo demostra os benefícios que a mudança pode alcançar em termos de aumento da taxa de transferência e redução das retransmissões e, também, alerta para o impacto que pode ocorrer no processo de verificação do canal. Os resultados contribuem com a literatura assistindo no avanço das redes locais sem fio da próxima geração.*

1. Introdução

A demanda por redes sem fio cresce rapidamente com a grande proliferação de novos dispositivos e aplicações. O número de aparelhos cujo principal acesso à rede é feito através de comunicação sem fio vem prevalecendo sobre a quantidade de aparelhos exclusivamente cabeados. Fatores como custo e simplicidade levaram à popularização em escala

mundial do padrão IEEE 802.11 utilizado nas WLANs, com mais de 7 bilhões de equipamentos em uso [Cisco 2014]. Entretanto, a saturação dos canais de comunicação destas redes tem sido observada nos grandes centros urbanos, principalmente aquelas operando na faixa livre de 2.4GHz. Isto resulta na degradação do desempenho e impacta profundamente na Qualidade de Experiência (QoE) dos usuários, requerendo novas soluções a fim de viabilizar **redes cada vez mais densas** [Alderfer 2013].

A dificuldade em prover uma boa qualidade em redes sem fio está no **compartilhamento do canal**. Pelo ar, os sinais emitidos podem sofrer influência de diversos tipos de ruídos e perturbações provenientes de outras fontes. Em ambientes com muitos dispositivos, há uma disputa para acessar ao meio, o qual precisa ser dividido entre todos os terminais desejando utilizá-lo. Caso aconteça alguma transmissão em paralelo na mesma faixa, haverá interferência entre elas e isto tornará ambos os sinais ilegíveis, causando descartes, desperdício de tempo e atrasos. Com isto, os protocolos de acesso ao meio tornam-se fundamentais no uso do canal de comunicação. Eles são os responsáveis por verificar se o canal está disponível ou ocupado, iniciando suas transmissões apenas quando disponível e abstendo-se de transmitir enquanto ocupado.

Em redes densas, a dificuldade em prover este acesso será maior por causa da quantidade de terminais disputando o meio. O atual mecanismo prevê uma aquisição aleatória do canal dentro de uma janela de tempo limitada. Com isto, a chance de uma transmissão simultânea cresce à medida que mais nós competem pelo canal. No sentido de melhorar este processo, a literatura tem apresentado diversas sugestões de mudança nas camadas física e de enlace [He et al. 2013]. Entretanto, as propostas que podem prover comunicação em duas vias, por múltiplos canais, exigem custos adicionais que inviabilizam sua interoperabilidade com os atuais equipamentos e introduzem novos problemas [Kosek-Szott 2012]. Em outra linha, sem a necessidade de novos canais, sugestões de mudanças nos parâmetros de tamanho da janela de contenção têm demonstrado capacidade de reduzir ou eliminar as colisões resultantes de transmissões simultâneas indesejadas. Contudo, estas opções esbarram no requisito de compatibilidade com o legado e na questão do gerenciamento das redes vizinhas, nas quais não é possível garantir cooperação.

Visando preencher estas duas lacunas, Hiertz et al. sugeriram alterar a duração dos *slots* no vigente processo de *backoff* de forma a manter as janelas com o mesmo tamanho em tempo, porém com mais opções de intervalos, reduzindo assim a chance de mais de um equipamento iniciar sua transmissão no mesmo instante e garantindo a compatibilidade com o legado [Hiertz et al. 2015]. A ideia foi apresentada ao grupo de trabalho TGax que atua na especificação da futura versão do padrão 802.11 voltado para redes densas, previsto para ser finalizado em 2019. Entretanto, esta solução não foi analisada seguindo o rigor da metodologia científica. Nosso trabalho contribui com a literatura provendo resultados de um estudo rigoroso da referida proposta, os quais oferecem direcionamentos importantes relacionados à redução das colisões e ao aumento da taxa de transferência.

O restante deste artigo está organizado como segue. A Seção 2 apresenta os trabalhos relacionados ao problema das colisões e a mudanças na duração dos slots. A Seção 3 detalha a proposta posicionando-a no contexto de redes densas. A Seção 4 descreve a avaliação de desempenho. A Seção 5 contempla o problema dos falsos positivos que podem ocorrer caso o equipamento não tenha tempo hábil para a verificação do canal. Por fim, a Seção 6 apresenta as conclusões e as oportunidades de trabalhos futuros.

2. Trabalhos Relacionados

Estudos práticos [Gupta et al. 2012, Patro et al. 2013] e analíticos [Cheng et al. 2013, Gottapu et al. 2013] mostram que o atual mecanismo de backoff apresenta altas taxas de colisões em cenários com grande número de estações e elevada carga de tráfego, resultando em baixas taxas de transferência. Buscando contornar esse problema, a literatura contém uma diversidade de pesquisas com sugestões para controlar dinamicamente o tamanho inicial da janela de contenção ou substituir todo mecanismo por outro livre de colisões. As propostas neste sentido podem ser classificadas em mecanismos baseados em divisão de tempo, canais adicionais de controle ou na regulagem dinâmica da janela. Entretanto, os mecanismos com base em divisão de tempo [He et al. 2013, Zheng et al. 2014] ou em canais de controle [Sen et al. 2011] esbarram no requisito de compatibilidade com os padrões atuais, pois seria muito difícil sua coexistência com os dispositivos já implantados e seriam exigidos recursos de banda que não estão disponíveis.

As propostas de adaptação dinâmica do tamanho da janela podem contribuir para um melhor desempenho, ajustando-a em tempo de execução conforme a quantidade observada de usuários concorrentes. Elas evitam as perdas que uma janela inicial muito grande geraria em um ambiente não saturado, assim como os problemas de uma janela pequena em um ambiente denso [Adnan e Park 2013]. Nesta perspectiva, as pesquisas têm sugerido mecanismos que adaptem o tamanho da janela através de um controle pelo ponto de acesso [Wang 2012] ou através de estimativas de uso, pela observação do tempo livre [Kang et al. 2010] ou do congestionamento do canal [Cheng et al. 2013]. Contudo, estas sugestões falham caso os equipamentos vizinhos não cooperem, como ocorrerá nos ambientes já implantados com o atual padrão. Em todos os casos, as pesquisas apontam que uma janela inicial maior é mais adequada para ambientes densos [Gottapu et al. 2013]. Entretanto, o aumento da janela criaria uma vantagem para os equipamentos legados que a mantivessem pequena.

A proposta avaliada neste trabalho difere das demais por aumentar a quantidade de slots da janela de contenção através da redução da duração dos slots. Com isto, o tamanho da janela é mantido com o mesmo intervalo total de tempo, evitando os atrasos que um intervalo maior geraria. Porém, a quantidade de slots aumenta, gerando mais oportunidades de transmissão e reduzindo a probabilidade de colisões [Hiertz et al. 2015]. De acordo com nosso conhecimento, as avaliações extensivas desta proposta não foram publicadas na literatura e há poucos estudos relacionados com mudanças na duração do slot em si. Entretanto, um pequeno número de resultados de pesquisa publicados permitenos avançar consequências preliminares sobre este parâmetro.

Em uma análise matemática de protocolos de acesso ao meio, [Rubinstein et al. 2007] demonstraram que o uso de slots curtos de 9μs do padrão 802.11g possui uma capacidade de evitar colisões em torno de 53% enquanto que a configuração com slots de 20μs possui apenas 43% desta capacidade nas mesmas condições. Em outro estudo, usando uma câmara de reverberação, [Primiani et al. 2010] realizaram medições para comparar equipamentos dos padrões 802.11b e 11g. O experimento mostrou que o tempo de resposta de conexões 11g (OFDM - 6Mbps) ficou em menos de um terço do que as 11b (CCK - 5Mbps) mesmo com altas taxas de retransmissões. Os autores explicaram o resultado através da diferença na duração do slot, curto no padrão 11g. Outro experimento realizado por [Harjula et al. 2011] avaliou interferências

externas em redes sem fio. Os resultados em *testbed* demonstraram que os protocolos com menor duração de slot conseguem resistir melhor às interferências por *jamming signals*. As conclusões obtidas revelam vantagens de um slot com menor duração.

Porém, outras pesquisas apontam que a redução do slot pode trazer inconvenientes. Em uma avaliação de sistemas de sensores baseados no padrão 802.11, os autores mediram a taxa de sucesso na transmissão de *beacons* em dispositivos 11g (20μs) e 11a (9μs) e perceberam que a taxa é menor neste último [Uchimura et al. 2010]. Eles justificaram que há menos oportunidades de transmissão para envio dos *beacons* porque o canal fica mais ocupado quando é utilizado um slot menor. Comparações entre os padrões 11n e 11ac identificaram problemas de coexistência relacionados com a duração do slot, como no estudo de [Abusubaih et al. 2013]. Isto ocorre porque todo o conjunto básico de serviços deve operar com slots de 20μs caso algum equipamento opere com uma duração diferente de 9μs. Estes trabalhos nos permitem prever algumas das consequências que a redução nos slots pode trazer, contudo, um intervalo de 4,5μs trará ainda outros desafios.

Com a redução para o intervalo de 4,5μs, os eventos que atualmente devem ocorrer dentro um slot com duração de 9μs precisam ser ajustados para que realizem suas funções em metade deste prazo, portanto, com influência no processo de verificação da disponibilidade do canal (CCA). A pesquisa de [Ramachandran e Roy 2007] investigou o impacto de imperfeições neste processo e calculou a influência de falsos positivos e falsos negativos no desempenho e na eficiência energética. O estudo revelou que em taxas de transferência maiores é mais importante contar com uma alta probabilidade de detecções corretas para evitar o custo das colisões. O foco, porém, estava em redes pessoais sem fio, com outro mecanismo de contenção e intervalos maiores que nas redes locais. Sobre o contexto das redes no padrão 802.11, nossa busca na literatura não encontrou estudos que esclareçam o quanto uma redução da duração do slot pode afetar o desempenho da detecção do canal ocupado.

3. Redução da Duração do Slot da Janela de Contenção

Em redes pouco saturadas, o mecanismo de acesso ao canal adotado pelo padrão 802.11 tem apresentado bons resultados, com adequado aproveitamento mesmo sob as altas taxas de transferência que foram adicionadas nas versões mais recentes [Chang et al. 2012]. Seu funcionamento é simples: quando percebem o canal disponível, as estações sorteiam um intervalo de tempo (slot) que determina o quanto devem esperar antes de iniciar sua transmissão, aquela que conseguir o menor intervalo iniciará primeiro e todas as demais se abstêm de transmitir enquanto o canal estiver ocupado. Esta contenção evita que todas as estações tentem acessar o canal ao mesmo tempo tão logo esteja livre. Porém, caso este sorteio resulte em um empate haverá uma transmissão simultânea, denominada colisão, com uma grande possibilidade de perda de ambos os quadros transmitidos.

A probabilidade de ocorrer uma colisão é inversamente proporcional à quantidade de slots que as estações têm para o sorteio na janela de contenção (CW) e diretamente proporcional ao número de estações competindo pelo acesso ao canal (n), conforme expressado em (1). Caso haja uma colisão, esta janela de tempo é aumentada de forma exponencial, reduzindo a possibilidade de uma nova colisão na próxima tentativa, até que a janela atinja o seu tamanho máximo. Porém, o desperdício do canal acontece a cada colisão, com impacto no desempenho, principalmente se as transmissões enviarem qua-

dros grandes. E, a cada transmissão bem-sucedida, o tamanho da janela retorna ao valor inicial, elevando a probabilidade de colisões independente do número de nós ou dos resultados anteriores. Há ainda outros tipos de colisões, produzidas pela presença de terminais ocultos, que iniciam suas transmissões porque não conseguem perceber o canal ocupado. Estas anomalias demandam outros tipos de solução que não são objetos deste estudo.

$$1 - \sum_{i=1}^{CW-1} \frac{n * (CW - i)^{(n-1)}}{CW^n} \tag{1}$$

Focando apenas nas colisões normais e inerentes ao funcionamento do protocolo, o impacto deste empecilho é mais significativo à medida que a quantidade de dispositivos aumenta. Com efeito, o problema será mais relevante em **redes densas**. Isto advém de uma janela inicial muito pequena para comportar o montante de aparelhos que demandam acesso ao meio [Gottapu et al. 2013]. As versões atuais do padrão 802.11 determinam uma janela inicial com apenas dezesseis intervalos, o que terá uma grande chance de empate em um sorteio com dez ou mais terminais competindo pelo acesso. E, no caso de tráfego priorizado de vídeo e voz (EDCA) esta janela é reduzida para oito e quatro intervalos respectivamente, aumentando ainda mais a chance de falhas se muitos equipamentos utilizarem estes serviços [IEEE 2012]. Neste caso, haverá uma probabilidade de colisões acima de 50% em uma rede com apenas 5 estações transmitindo quadros de voz no mesmo canal e provavelmente haverá descartes, impactando fortemente na percepção de qualidade pelo usuário.

Contudo, conjuntos com dezenas ou centenas de dispositivos operando em um único ponto de acesso podem ser encontrados com facilidade. Locais públicos como estádios, estações de trem, aeroportos, escritórios corporativos, ambientes acadêmicos e outros com grande aglomeração de pessoas têm uma alta probabilidade de conterem esta quantidade de usuários em seus *hotspots* de acessos WiFi. Além do mais, considerando que redes próximas operando na mesma faixa de frequência dividem o mesmo espaço, redes vizinhas com pontos de acesso diferentes podem formar agrupamentos de muitos dispositivos operando no mesmo canal, situação comum em condomínios de prédios residenciais e comerciais. Sendo assim, as próximas gerações de redes sem fio devem considerar melhorias que permitam atender aos requisitos para esta quantidade de usuários [Deng et al. 2014].

Visando aperfeiçoar os instrumentos de controle das camadas física e de enlace das redes locais sem fio do padrão 802.11, o IEEE criou o grupo de estudos denominado *High Efficiency Wireless LAN*, também conhecido como *Task Group AX* ou TGax. A proposta de alargamento da janela de contenção pela redução dos slots foi aprovada pelo TGax para análise em avaliação preliminar e aguarda pelas fases de revisão. O principal objetivo desta agremiação é aumentar a taxa de transferência e a eficiência no aproveitamento do espectro em ambientes com elevada quantidade de nós. Entretanto, a nova emenda que está sendo aperfeiçoada pelo grupo deve manter a compatibilidade com os dispositivos legados e permitir sua coexistência quando operando nas mesmas faixas de frequência [Wang et al. 2014].

3.1. Detalhamento da Proposta

A proposta apresentada em [Hiertz et al. 2015] sugere uma **redução na duração do slot na proporção inversa do crescimento da janela**, aumentando a quantidade de intervalos da janela inicial de 16 para 32 slots e diminuindo o slot de 9μs para 4,5μs, conforme ilustrado pela Figura 1. De forma similar, as janelas de tráfego prioritário devem ser ajustadas conforme as regras já existentes para definição do seu tamanho, mantendo os mesmos valores na duração das janelas de contenção, limitado em 1024 slots na janela máxima (Tabela 1). Assim, equipamentos novos permanecem compatíveis com os padrões atuais se os demais intervalos baseados no slot forem ajustados na mesma proporção. Além do mais, esta abordagem oferece uma pequena vantagem sobre os modelos anteriores por causa do maior número de possibilidades que terão no sorteio do *backoff*, incentivando a atualização com coexistência.

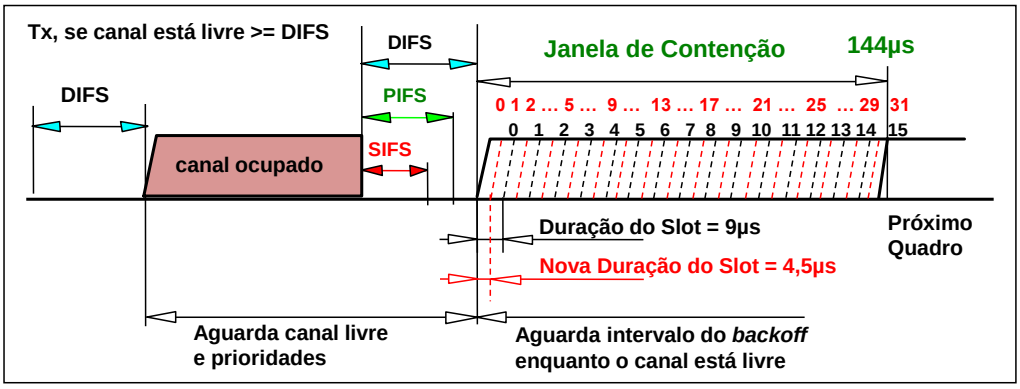


Figura 1. Alteração Proposta ao Processo de Backoff

Tabela 1. Parâmetros da Janela de Contenção (atual x proposto)

Categoria EDCA	Regra		Slot = 9μs			Slot = 4,5μs		
	Janela Inicial	Janela Máxima	CWmin (slots)	CWmin (μs)	CWmax (μs)	CWmin (slots)	CWmin (μs)	CWmax (μs)
AC_BK	CWmin	CWmax	16	144	9216	32	144	4608
AC_BE	CWmin	CWmax	16	144	9216	32	144	4608
AC_VI	CWmin/2	CWmin	8	72	144	16	72	144
AC_VO	CWmin/4	CWmin/2	4	36	72	8	36	72

Equipamentos operando com slots de 9μs ou 4,5μs podem coexistir no mesmo ambiente e com o mesmo processo de acesso ao canal se as janelas de contenção forem mantidas com o intervalo total de tempo inalterado. Neste caso, não ocorrem os inconvenientes relatados em [Abusubaih et al. 2013] quando todo o conjunto básico de serviços tem que operar com o maior slot por causa de algum terminal legado operando no mesmo conjunto. Para isto, os intervalos baseados na duração do slot precisam ser ajustados de forma a ficarem com o mesmo espaço de tempo, entre estes: AIFS (aSlotTime + SIFS $\rightarrow$ aSlotTime * 2 + SIFS) e CWmin (15 $\rightarrow$ 31). Desta maneira, os equipamentos novos passam a ter uma probabilidade menor de que suas transmissões resultem em colisões por causa do maior número de slots para iniciarem seus acessos e, por consequência, há redução em retransmissões e descartes de quadros.

4. Avaliação de Desempenho

Modelos matemáticos têm sido amplamente utilizados para caracterizar o comportamento do processo de *backoff*. Vários modelos foram apresentados e refinados em dezenas de trabalhos que se diferenciaram por incluir alguma nova variável do mecanismo. Porém, não há consenso sobre uma equação de ponto fixo que determine a probabilidade de colisões em todos os cenários [Dai e Sun 2013]. Sendo assim, optamos por avaliar a proposta através de simulações, descritas nesta seção, que nos permitem obter dados mais concretos com diferentes tipos de tráfego e contextos.

4.1. Metodologia

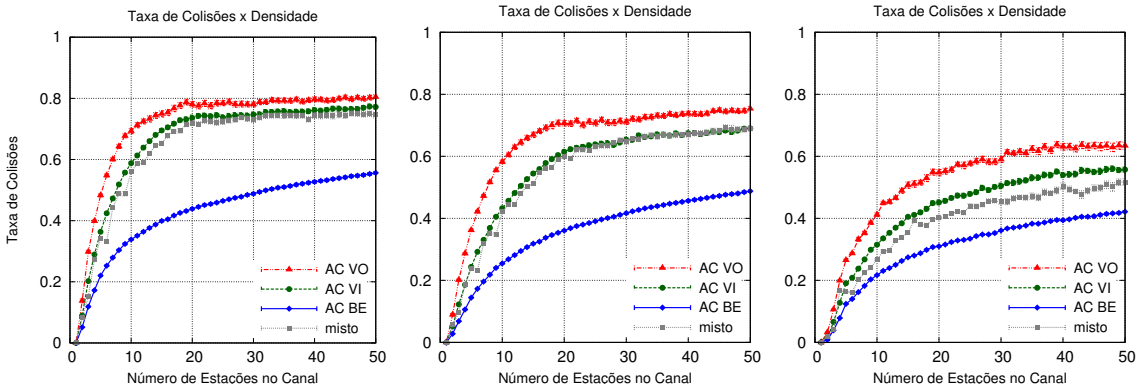
Utilizamos o simulador de redes *ns3* e implementamos um cenário com dois pontos de acesso compartilhando o mesmo canal, afastados em 20 metros, com estações uniformemente distribuídas a 10 metros dos seus pontos de acesso. O padrão utilizado foi o 802.11n, base para o 802.11ax, com modulação e esquema de codificação MCS7 (64-QAM) para dados, MCS0 (BPSK) para controle na camada física e intervalos de envio de *beacons* em 100ms, como é característico neste tipo de rede. As estações enviaram fluxos de dados UDP para suas bases com uma geração de tráfego do tipo ON/OFF em períodos de 50ms em média, de acordo com uma distribuição de Pareto, similar aos modelos que representam o tráfego Web típico predominante em *hotspots* públicos [Divgi e Chlebus 2013]. As funções de proteção RTS/CTS foram desabilitadas, o tamanho dos pacotes foi constante, com 1000 ou 256 bytes, as filas foram ajustadas em 1000 quadros e os demais parâmetros permaneceram com o valor padrão.

A quantidade de estações variou a cada amostragem com duração de 10 segundos, iniciando com apenas um terminal e encerrando com 50, repetindo este processo 40 vezes, reiniciado os contadores e o gerador de números aleatórios do simulador a cada amostra. Realizamos o experimento com diferentes tipos de tráfego, que foi marcado como voz (AC_VO), vídeo (AC_VI) e não prioritário (AC_BE). Utilizamos, também, uma opção mista em que cada estação enviou quadros de um destes três tipos. O experimento foi realizado com diferentes cenários: com todas as estações operando com slots de 9μs e janelas conforme o padrão atual, todas operando com slots de 4,5μs e janelas alargadas da forma como está sendo proposto para o 802.11ax e um terceiro cenário com metade das estações em cada uma destas opções. Sobre o volume de tráfego gerado, as análises foram realizadas de duas formas: em um contexto de canal saturado, com cada estação gerando 54Mbps e outro não-saturado, com o total de tráfego no canal em 54Mbps.

Obtivemos a quantidade de colisões e calculamos a **taxa de colisões** em relação ao total de quadros recebidos através de um detector de colisões que inserimos no simulador. Este recurso permitiu contar os quadros recebidos enquanto o nó já estava em recepção, identificando cada colisão de quadros. Este indicador ficou bastante próximo da **taxa de erros de transmissão**, obtida pela quantidade de quadros enviados sem o retorno da respectiva confirmação (ACK). A diferença pode ser explicada pela colisão de mais de dois quadros. A **taxa de transferência** foi auferida pela quantidade de bytes recebidos pela aplicação em seu destino. Os parâmetros de tamanho das janelas, duração dos slots e os intervalos para priorização entre quadros (AIFS) foram ajustados em cada fila da função de coordenação distribuída (DCF) e do controle de acesso distribuído aprimorado (EDCA) para os cenários da proposta com slots de 4,5μs.

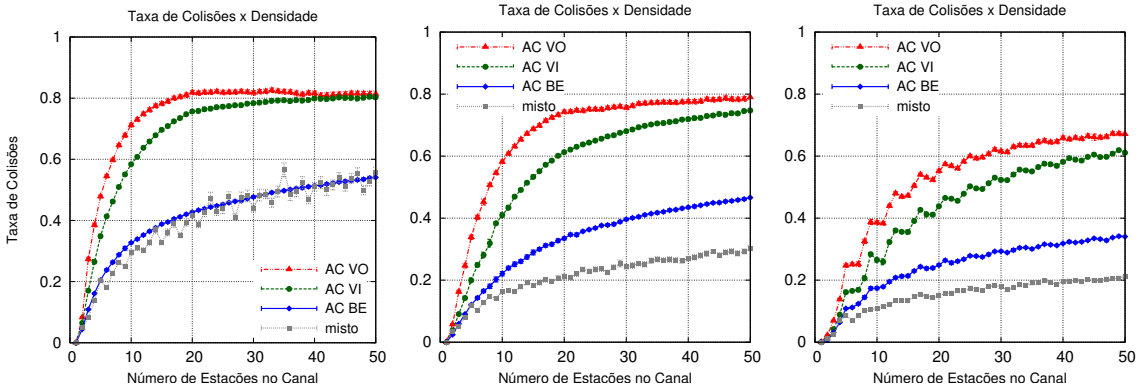
4.2. Resultados

As taxas de colisões foram reduzidas com o alargamento das janelas conforme previsto. O resultado foi mais significativo para o tráfego prioritário, principalmente no contexto do canal saturado conforme ilustrado na Figura 2. O tráfego não prioritário já contava com uma janela inicial maior e aumentos exponenciais com mais iterações, tendo resultados mais expressivos com o canal pouco saturado conforme visto na Figura 3. Em ambos os casos, a redução de 9,2 para 4,6 segundos que ocorreu no tamanho máximo da janela não comprometeu o resultado (CWmax - Tabela 1). O efeito foi melhor no ambiente misto por causa da vantagem obtida pelos nós operando com slots menores, deixando de concorrer com o legado em metade das oportunidades de acesso que tiveram (Figuras 2(c) e 3(c)). Contudo, em todos os ambientes e tipos de tráfego podemos constatar uma curva mais suave na taxa de colisões, aumentando menos conforme o crescimento da densidade.



(a) 9μs (b) 4,5μs (c) misto

Figura 2. Canal Saturado: Taxa de Colisões x Densidade da Rede

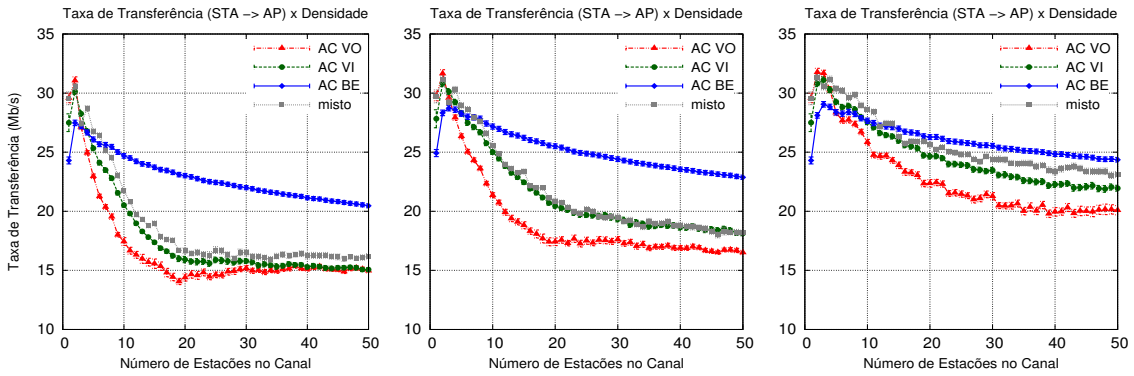


(a) 9μs (b) 4,5μs (c) misto

Figura 3. Canal não Saturado: Taxa de Colisões x Densidade da Rede

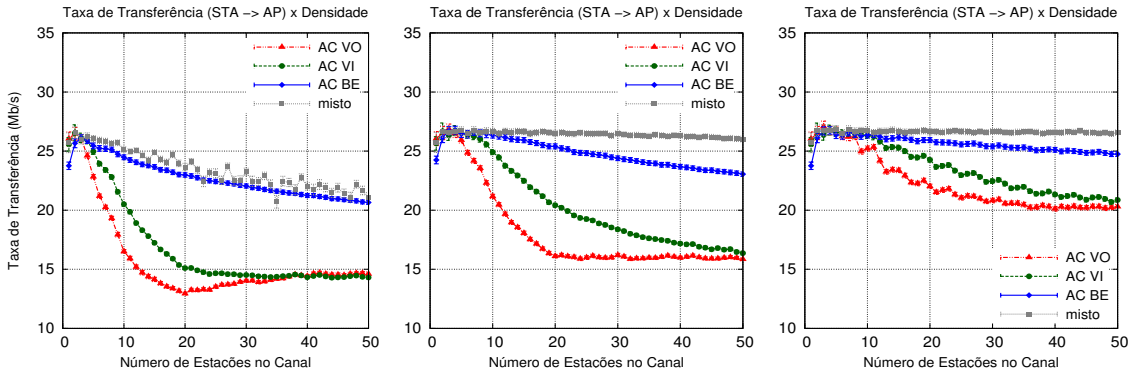
As taxas de transferência refletiram a redução na taxa de colisões. O resultado foi mais expressivo nos perfis de tráfego prioritário, que passou a apresentar uma curva mais próxima do conteúdo não prioritário, com melhor rendimento conforme ilustrado pelas Figuras 4 e 5. A perda que normalmente ocorre ao utilizar uma janela inicial maior nas

transmissões com poucos nós concorrentes foi eliminada pelo uso do slot menor. E, todos os tipos de tráfego apresentaram aumento na vazão, novamente com resultados mais significativos em ambiente misto, tanto no contexto do canal saturado quanto não saturado. Sendo assim, podemos concluir que o aumento da taxa de transferência foi alcançado com o menor número de retransmissões resultante da diminuição das colisões. A redução na quantidade de erros de transmissão, conforme ilustrado na Figura 6, corrobora com esta dedução. Isto denota a melhora na eficiência do aproveitamento do espectro, com mais oportunidades de envio e menos desperdício do tempo do canal.



(a) 9μs (b) 4,5μs (c) misto

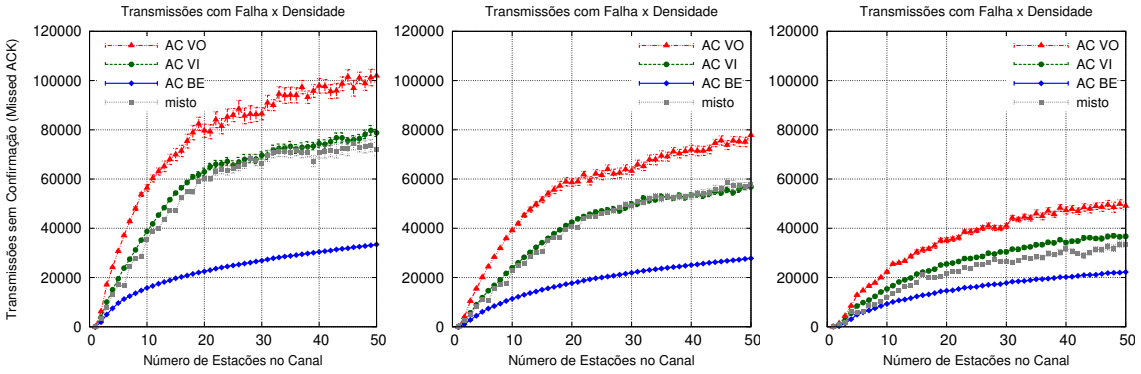
Figura 4. Canal Saturado: Taxa de Transferência x Densidade da Rede



(a) 9μs (b) 4,5μs (c) misto

Figura 5. Canal não Saturado: Taxa de Transferência x Densidade da Rede

O comportamento foi bastante parecido independente do tamanho dos quadros. As transferências com pacotes menores, de 256 bytes, apresentaram maiores taxas de colisão tanto em slots de 9μs quanto 4,5μs, com pequena variação em tráfego não prioritário. Sendo assim optamos pelos exemplares com pacotes de 1000 bytes nas figuras desta exposição. Obviamente, a taxa de transferência será superior com quadros maiores ou agregados, pois estes exigem menos quadros para obter a mesma vazão e, consequentemente, menos colisões. Quanto ao tamanho das amostras, as medições com intervalos de 10, 5 ou 1 segundo não afetaram de modo expressivo o produto, apenas suavizaram as curvas, mas mantiveram as mesmas tendências. Em todas estas variações, os resultados foram mais expressivos nas amostras em ambiente misto.



(a) 9μs (b) 4,5μs (c) misto

Figura 6. Quantidade de Erros de Transmissão x Densidade da Rede

Com isto, vislumbramos a possibilidade de outras aplicações para estes novos slots, como por exemplo, a reserva de alguns deles para tráfego prioritário ou para outro mecanismo de acesso, similar ao proposto em [He et al. 2013] com a uma nova regra de distribuição dos slots nos quais determinadas estações possam tentar o acesso. Com efeito, entendemos que o momento é oportuno para inclusão de novas estruturas. Os equipamentos operando com slots menores podem com contar essa vantagem no acesso ao canal somente até que todos os dispositivos sejam substituídos. Mas, caso todos estejam operando com o mesmo tamanho de slot e janela, os ganhos desta proposta não são tão significativos, sendo que novas reduções de slots podem ser inviáveis.

5. Análise Crítica

Apesar dos resultados positivos, a viabilidade da redução do slot deve ser alcançada antes que seja possível contar com este recurso. A versão 2012 do padrão 802.11 determina a relação de tempos tolerados para cada processamento dentro do intervalo de um slot conforme o item 9.3.7 (Figura 7). Os valores podem variar conforme a modulação utilizada, mas as tarefas de processamento da camada de enlace e de troca do estado de RX para TX devem ocorrer em menos de 2μs cada. E a verificação do canal deve ocorrer em menos de 4μs quando operando em modulações que utilizam slots de 9μs, como, por exemplo, em multiplexação por divisão de frequência ortogonal (OFDM) em canais de 20MHz ou quando operando com múltiplas antenas (MIMO) (Tabela 2). O processo de verificação do canal pode ser feito pela detecção do preâmbulo de um quadro, que exige mais tempo, ou pela medição do nível de energia do canal, processo mais rápido, porém menos preciso, detalhados mais adiante.

Tabela 2. Duração Máxima de cada Procedimento no Slot

	OFDM PHY			**MIMO PHY**		**ERP-OFDM**		HR PHY	DS PHY	IR PHY	FH PHY
Característica	**20**MHz	**10**MHz	**5**MHz	short	long	short	long				
aSlotTime	**9** μs	**13** μs	**21** μs	**9** μs	**20** μs	**9** μs	**20** μs	**20** μs	**20** μs	**8** μs	**50** μs
aCCATime	**<4** μs	**<8** μs	**<16** μs	**<4** μs	**<4** μs	**<4** μs	**<15** μs	$\leqslant$ **15** μs	$\leqslant$ **15** μs	**5** μs	**27** μs
Troca Rx/Tx	**<2** μs	**<2** μs	**<2** μs	**<2** μs	**<2** μs	**<5** μs	**<5** μs	$\leqslant$ **5** μs	$\leqslant$ **5** μs	**0** μs	20 μs
Atraso MAC	**<2** μs	**<2** μs	**<2** μs	**<2** μs	**<2** μs	**<2** μs	**<2** μs	$\leqslant$ **2** μs	$\leqslant$ **2** μs	**2** μs	**2** μs

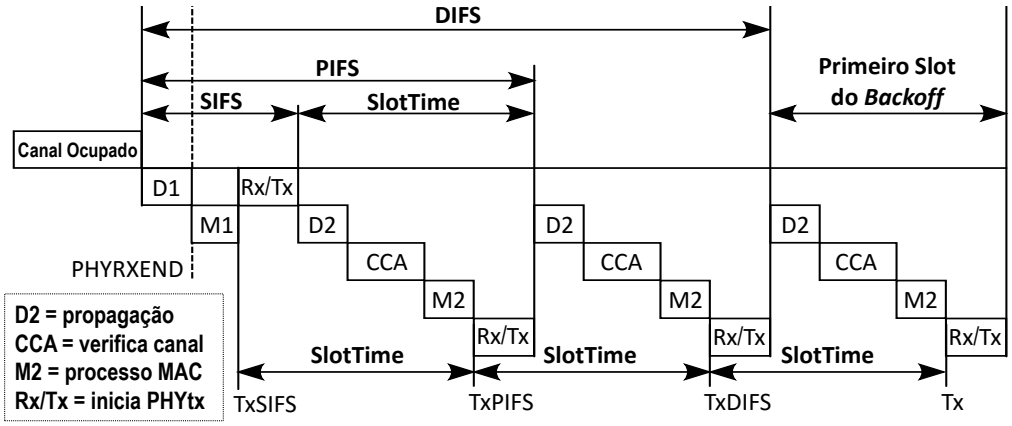
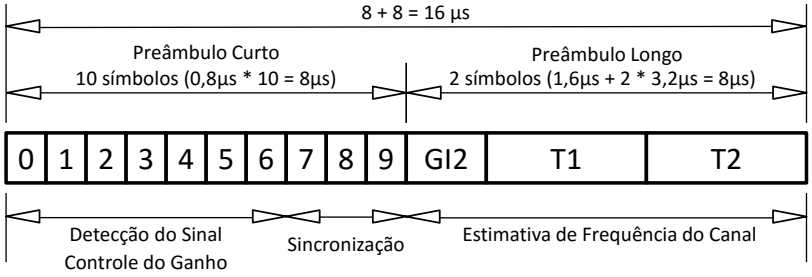


Figura 7. Relação de Tempos do Processo DCF [IEEE 2012]

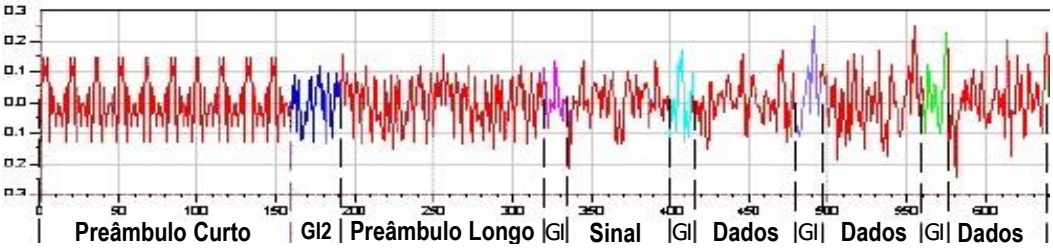
O desafio a ser superado para viabilizar a redução dos slots é a execução destes processos em um intervalo de apenas 4,5μs. Como o tempo de propagação é constante (1μs para percorrer 300m), os demais processos que hoje demandam um intervalo de 8μs terão que ser realizados em apenas 3,5μs. Hiertz e os demais autores consideram que as implementações atuais destes processos demandam muito menos tempo do que quando foram inicialmente padronizadas. Sendo assim, do ponto de vista deles, a evolução do *hardware* pode entregar estas funções no intervalo proposto [Hiertz et al. 2015]. Havendo esta viabilidade, a probabilidade de colisões será reduzida, minimizando o problema de uma maneira pouco intrusiva e permitindo a compatibilidade e a coexistência com os equipamentos legados que operarem com slots de 9μs.

Enquanto inspecionam o canal, para efetuar a sincronização com as transmissões em curso, os nós devem efetuar a detecção do preâmbulo do pacote, tipicamente consistindo de uma sequência repetitiva de símbolos conhecidos. Nos quadros usando modulação OFDM, o cabeçalho contém uma sequência curta de símbolos destinados a detecção e sincronização, ocupando 8μs do preâmbulo (Figura 8). O processo de verificação de canal deve definir se o canal está ocupado utilizando apenas cinco dos dez símbolos do campo, em um intervalo de 4μs quando operando em 20MHz [IEEE 2012]. E, em um slot reduzido pela metade, este mesmo processo terá que ser feito com apenas dois símbolos. Contudo, caso o *hardware* seja capaz de reconhecer cada um destes símbolos com precisão, apenas um é suficiente e, uma vez detectado, o nó entraria em processo de recepção e interromperia o *backoff* imediatamente.

Não obstante, no item 18.3.10.6 da vigente emenda do padrão 802.11, há outro método para definir se o canal está em uso quando a detecção do preambulo falha ou não é possível, como no caso do canal estar ativo por transmissões em outras modulações não suportadas pelo nó. Este item define o método de verificação do canal pela detecção de energia, considerando-o ocupado caso a energia medida esteja 20dB acima da sensibilidade mínima para a modulação e taxa de codificação. Com isto, o mecanismo conta com duas formas de tomar a decisão. E, como o tempo para efetuar uma simples medição da energia no canal é relativamente curto, entendemos que a verificação da disponibilidade do meio pode ser viável com um intervalo de slot reduzido. Todavia, sua execução depende das implementações do *hardware*.



(a) Modelo lógico



(b) Representação do Sinal

Figura 8. Preâmbulo de um Quadro OFDM

A emenda atual do padrão 802.11, no item 19.4.7, exige uma eficácia de 99% para o processo de verificação do canal quando utilizando slots de 20μs. Este mesmo índice é tolerado em 90% nas modulações que utilizam o slot menor, com duração de 9μs (Figura 3). Obviamente, a redução do intervalo de medição para detecção do canal ocupado deve influenciar a capacidade do *hardware* de entregar medições precisas. Porém, os processadores evoluíram extraordinariamente desde 1999 quando estes parâmetros foram estabelecidos na emenda 802.11a para que o padrão suportasse a multiplexação por divisão de frequência ortogonal (OFDM).

Tabela 3. Desempenho na Verificação do Canal [IEEE 2012]

Parâmetro	Duração do Slot = 20μs	Duração do Slot = 9μs
SlotTime	20μs	9μs
RxTxTurnaroundTime	5μs	5μs
CCA_Time	15μs	4μs
CCA_Detect_Probability	>99%	>90%

O estudo das imperfeições no processo de verificação do canal e suas consequências é o próximo passo para confirmar a viabilidade da proposta caso o *hardware* não consiga entregar detecções precisas em um intervalo tão curto. Se alguma possível falha gera falsos positivos, identificando o canal ocupado quando de fato está disponível, há desperdício de tempo, mas não uma colisão. Entretanto, se as falhas geram falsos negativos, o canal é considerado livre quando de fato está ocupado e consequentemente há uma colisão. Porém, os principais simuladores de redes não oferecem modelos probabilísticos para o processo de verificação do canal. Conforme descrito em [Kai e Liew 2010] os modelos utilizam uma relação binária que não reflete o processo observado em experimentos. Com isto, há uma lacuna que pretendemos explorar em nossos próximos trabalhos.

6. Conclusão

O presente artigo discutiu a proposta de alargamento do tamanho inicial da janela de contenção através da redução da duração dos slots que está sendo avaliada para o 802.11ax. Nosso trabalho avaliou a mudança e confirmou os resultados previstos quanto à redução das colisões e aumento da vazão, efetuando ensaios com diferentes tipos de tráfego, variando em tamanho e nas categorias de priorização. Nosso estudo mediu as consequências do sugerido plano em aspectos relevantes como as taxas de erros e de transferência e ponderou os resultados, identificando ganhos e oportunidades. Através de extensivas simulações nosso trabalho confirmou que o uso de janelas maiores aumenta a vazão e reduz os erros de transmissão. De forma similar, demonstramos que a redução do slot conforme proposta garante a manutenção das taxas de transferências em cenários com poucos nós e, também, a compatibilidade e a coexistência com os equipamentos legados.

Constatamos que os novos intervalos criados no processo de alargamento das janelas pela redução dos slots podem ser utilizados para inserir mudanças no método de acesso. Isto inicia um novo direcionamento de pesquisas, que utilizem estes novos slots de forma diferenciada para priorização de tráfego ou reservas de canal controladas pelo ponto de acesso ou outro mecanismo. Também alertamos que o uso de intervalos menores terá impacto no processo de verificação do canal, aumentando a possibilidade de falsos negativos que podem gerar novos tipos de colisão caso o *hardware* não seja capaz de efetuar detecções precisas na avaliação do canal que pretende acessar. Esperamos, com isto, ter contribuído para um importante passo na exploração do ambiente que está sendo planejado para as próximas gerações de redes locais sem fio.

Referências

Abusubaih, M. A., Najem Eddin, S., e Khamayseh, A. (2013). IEEE 802.11n Dual Band Access Points for Boosting the Performance of Heterogeneous WiFi Networks. In *ACM PM2HW2N*, pages 1–4.

Adnan, M. e Park, E.-C. (2013). Assuring Per-Station Fairness in Multi-Rate WLANs: A Hybrid Approach of Contention Window Control and Frame Aggregation. In *ICUFN*, pages 282–287.

Alderfer, R. (2013). Wi-Fi Spectrum: Exhaust Looms. *Disponível em SSRN 2411645*. URL: http://ssrn.com/abstract=2411645 Último Acesso: 27/12/2015.

Chang, Z., Alanen, O., Huovinen, T., Nihtilä, T., Ong, E. H., Kneckt, J., e Ristaniemi, T. (2012). Performance Analysis of IEEE 802.11ac DCF with Hidden Nodes. In *IEEE VTC Spring*, pages 1–5.

Cheng, M.-H., Hwang, W.-S., Lin, C.-H., e Su, H.-K. (2013). A Oneself Adjusts Backoff Mechanism for Channel Access in IEEE 802.11 DCF WLAN. In *CISIS*, pages 287–292.

Cisco, C. V. N. (2014). Global Mobile Data Traffic Forecast Update 2014–2019. White Paper c11-520862. *Disponível em http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf*. Último acesso: 27/12/2015.

Dai, L. e Sun, X. (2013). A Unified Analysis of IEEE 802.11 DCF Networks: Stability, Throughput, and Delay. *IEEE Trans. Mobile Comput.*, 12(8):1558–1572.

Deng, D.-J., Chen, K.-C., e Cheng, R.-S. (2014). IEEE 802.11ax: Next Generation Wireless Local Area Networks. In *QSHINE*, pages 77–82.

Divgi, G. e Chlebus, E. (2013). Characterization of User Activity and Traffic in a Commercial Nationwide Wi-Fi Hotspot Network: Global and Individual Metrics. *Wireless Networks*, 19(7):1783–1805.

Gottapu, S. B. R., Tatineni, M., e Kumar, M. (2013). Performance Analysis of Collision Alleviating Distributed Coordination Function Protocol in Congested Wireless Networks - A Markov Chain Analysis. *IET Networks*, 2(4):204–213.

Gupta, A., Min, J., e Rhee, I. (2012). WiFox: Scaling WiFi Performance for Large Audience Environments. In *ACM CoNEXT*, pages 217–228.

Harjula, I., Pinola, J., e Prokkola, J. (2011). Performance of IEEE 802.11 Based WLAN Devices Under Various Jamming Signals. In *IEEE MILCOM*, pages 2129–2135.

He, Y., Sun, J., Ma, X., Vasilakos, A. V., Yuan, R., e Gong, W. (2013). Semi-Random Backoff: Towards Resource Reservation for Channel Access in Wireless LANs. *IEEE/ACM TON*, 21(1):204–217.

Hiertz, G., Mestanov, F., e Coffey, S. (2015). Enlarged Minimal Contention Window Size. Technical report, doc: IEEE 802.11-15/0914r1. URL: https://mentor.ieee.org/802.11/dcn/15/11-15-0914-01-00ax-enlarged-minimal-contention-window-size.pptx Último Acesso: 27/12/2015.

IEEE (2012). 802.11-2012 - IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical Report IEEE Std 802.11-2012, IEEE-Inst.

Kai, C. H. e Liew, S. C. (2010). Towards a More Accurate Carrier Sensing Model for CSMA Wireless Networks. In *IEEE ICC*, pages 1–6.

Kang, S.-W., Cha, J.-R., e Kim, J.-H. (2010). A Novel Estimation-Based Backoff Algorithm in the IEEE 802.11 Based Wireless Network. In *IEEE CCNC*, pages 1–5.

Kosek-Szott, K. (2012). A Survey of MAC Layer Solutions to the Hidden Node Problem in Ad-Hoc Networks. *Ad Hoc Networks*, 10(3):635–660.

Patro, A., Govindan, S., e Banerjee, S. (2013). Observing Home Wireless Experience Through WiFi APs. In *ACM MobiCom*, pages 339–350.

Primiani, V. M., Moglie, F., e Recanatini, R. (2010). On the Use of a Reverberation Chamber to Test the Performance and the Immunity of a WLAN System. In *IEEE EMC 2010*, pages 668–673.

Ramachandran, I. e Roy, S. (2007). Clear Channel Assessment in Energyconstrained Wideband Wireless Networks. *IEEE Wireless Commun.*, 14(3):70–78.

Rubinstein, M. G., Costa, L. H. M. K., Campista, M. E. M., de Oliveira Cunha, D., Amodei Jr, A., Velloso, P. B., e Duarte, O. C. M. (2007). Analysis of MAC Protocols for Home Networks. *Journal of Commun. and Inform. Sys.*, 22(1):10–23.

Sen, S., Roy Choudhury, R., e Nelakuditi, S. (2011). No Time to Countdown: Migrating Backoff to the Frequency Domain. In *ACM MobiCom*, pages 241–252.

Uchimura, Y., Nasu, T., e Takahashi, M. (2010). IEEE 802.11-Based Wireless Sensor System for Vibration Measurement. *Advances in Civil Engineering*, 2010.

Wang, C. (2012). Achieving Per-Flow and Weighted Fairness for Uplink and Downlink in IEEE 802.11 WLANs. *EURASIP Journal on Wireless Commun. and Networking*, 2012(1):1–9.

Wang, L., Lou, H.-L., Zhang, H., Sun, Y., Chu, L., Lan, Z., Zhang, J., Kang, H., Chang, S., e Taori, R. (2014). Proposed 802.11ax Functional Requirements. Technical report, doc: IEEE 802.11-14/0567r3. URL: https://mentor.ieee.org/802.11/dcn/14/11-14-0567-03-00ax-proposed-tgax-functional-requirements.doc Último Acesso: 27/12/2015.

Zheng, L., Ni, M., Cai, L., Pan, J., Ghosh, C., e Doppler, K. (2014). Performance Analysis of Group-Synchronized DCF for Dense IEEE 802.11 Networks. *IEEE Trans. Wireless Commun.*, 13(11):6180–6192.

SDMA Group Selection for Reduced Computational Complexity on MU MIMO Systems

László R. Costa[12], F. Rafael M. Lima[12], Tarcisio F. Maciel[1], Yuri C. B. Silva[1], F. Rodrigo P. Cavalcanti[1]

[1] Telecomunnications Research Group (GTEL),
Federal University of Ceará, Fortaleza, Brazil

[2]Programa de Pós-Graduação em Engenharia Elétrica e de Computação (PPGEEC),
Federal University of Ceará, Sobral, Brazil

{laszlon,rafaelm,maciel,yuri,rodrigo}@gtel.ufc.br

Abstract. *Optimizing wireless systems with Multiple Input Multiple Output (MIMO) by means of Radio Resource Allocation (RRA) in general requires very complex algorithms. In this work, we revisit two Radio Resource Allocation (RRA) problems in this scenario: Unconstrained Rate Maximization (URM) and Constrained Rate Maximization (CRM). In order to optimally solve those problems it is necessary to list all possible Space Division Multiple Access (SDMA) groups combinations and calculate the transmit and receive filters of the terminals for each combination. The main proposal in this work is to decrease the complexity of the RRA solutions at the cost of a controlled performance degradation by selecting only a fraction of all possible SDMA groups.*

1. Introduction

The advantages of Multiple Input Multiple Output (MIMO) technology over single antenna systems have become apparent to wireless research community mainly after the seminal works of Foschini [Foschini 1996] and Alamouti [Alamouti 1998] at the end of 1990's. Almost twenty five years later, MIMO technology have become mandatory in modern communication standards such as Long Term Evolution (LTE)-Advanced and Worldwide Interoperability for Microwave Access (WiMAX) [Li et al. 2010].

The MIMO technology together with Orthogonal Frequency Division Multiple Access (OFDMA) can be used to obtain gains over single antenna systems in transmit data rate, bit error rates and co-channel interference mitigation [Mietzner et al. 2009]. Multiuser (MU) MIMO schemes, also known as Space Division Multiple Access (SDMA), consist in the use of multiple antennas to enable the allocation of different spatial subchannels to different terminals in the same time-frequency resource [Gesbert et al. 2007]. In this case, the spatial dimension can be used as another tool to exploit the multiuser diversity.

The joint use of MIMO and Radio Resource Allocation (RRA) is a relevant strategy to deal with the challenges of next generation of cellular networks. However, the MU MIMO capability turns the RRA problems even more challenging due to the added degree of freedom and, therefore, requires computationally expensive solutions. In this work we revisit two RRA problems in MU MIMO scenario: Unconstrained Rate Maximization (URM) and Constrained Rate Maximization (CRM) [Lima et al. 2014]. An

important problem in the literature is the total data rate (sum rate) maximization or URM. The achievable sum rate capacity for the multi-antenna downlink channel has been found in [Caire and Shamai 2003] while [Tejera et al. 2006] shows how that sum rate capacity can be found using a non-linear processing technique called Dirty Paper Coding (DPC) [Costa 1983]. However, obtaining the optimal transmission policy when employing DPC is a computationally complex non-convex problem. Motivated by this, linear processing at the transceivers has been adopted due to the good achieved performance-complexity trade off [Ho and Liang 2009]. The Block Diagonalization (BD)-Zero Forcing (ZF) strategy is a linear filtering scheme capable of spatially multiplexing multiple terminals in the same frequency resource in a MU-MIMO [Spencer et al. 2004]. As will be shown later, in order to obtain the optimal solution of the URM problem when linear spatial filtering is used, we need to calculate the transmit and receive filters of the terminals of all possible SDMA group combinations in all available frequency resources.

Although URM solution achieves the maximum spectral efficiency, it is unfair which leads to service starvation to terminals with unfavorable channel conditions. The total data rate maximization subject to satisfaction constraints or CRM has been considered in [Lima et al. 2014]. The objective of CRM is the same as the URM problem, however, the former assumes a multiservice scenario where each terminal is assumed to be transmitting or receiving data from a multimedia service, e.g., Voice over IP (VoIP) or web browsing. Therefore, besides maximizing the total transmit data rate, the resource assignment should guarantee that a minimum number of terminals of each service is satisfied with the provided Quality of Service (QoS). In order to obtain the optimal solution of CRM problem, the authors have shown that the original non-linear optimization problem can be converted to an Integer Linear Program (ILP) and then solved by Branch-and-Bound (BB)-based algorithms [Nemhauser and Wosley 1999]. However, as in the URM problem, the linear spatial filters at the transmitter and receiver of all terminals in all possible SDMA groups and frequency resources should be pre-calculated. This operation is computationally intense for both URM and CRM problems.

The main contribution of this article is the proposal of methods to pre-select a fraction of all possible SDMA groups based on simple and reasonable metrics reducing the problem size proposed in [Lima et al. 2014]. The selected SDMA groups will be used in the solutions of the URM and CRM problems. With this approach, only the spatial filters for the terminals of the selected SDMA groups should be calculated. Besides saving the computational task of calculating filters of terminals in some SDMA groups, the solutions of the URM and CRM problems will have their computational complexity decreased. The main contributions of this work are: Proposal of a framework to reduce the computational complexity to obtain the optimal solutions of problems URM and CRM; Proposal of metrics to measure the relevance of SDMA groups; Performance evaluation of the proposed framework and grouping metrics; Calculation of the computational complexity of the solutions to URM and CRM problems with our proposed framework.

The remainder of this article is organized as follows. In section 2, we provide the system model and main assumptions. The proposed solution to decrease the computational complexity is presented in section 3 and performance results based on simulations are shown in section 4. Finally, the main conclusions and perspectives are depicted in sections 5.

2. System Modeling

2.1. Definitions and variables

Most of the system scenario and variables are similar to the ones presented in [Lima et al. 2014]. Basically, we assume RRA in the downlink of a given cell of an OFDMA cellular system. The minimum allocable resource, or Resource Block (RB), consists in a group of one or more adjacent subcarriers and a number of consecutive Orthogonal Frequency Division Multiplexing (OFDM) symbols in the time domain, which represents the Transmission Time Interval (TTI). We assume a MU MIMO scenario with M_T and M_R antennas at the Base Station (BS) and terminals, respectively. There are J terminals in the considered cell and N available RBs. $\mathcal{J}$ and $\mathcal{N}$ are the set of available terminals and RBs, respectively. As the CRM problem assumes a multiservice scenario, we consider that each terminal is using a service $s \in \mathcal{S}$, e.g., web browsing or file download, where $\mathcal{S}$ is the set of all services. The set of terminals using service s is given by $\mathcal{J}_s$. The number of terminals from service s is $|\mathcal{J}_s| = J_s$.

The MIMO channel of the link between the BS and terminal j on RB n is modeled by the matrix $H_{j,n}$ that is composed of the elements $h_{j,n,a,b}$ that represents the channel transfer function between the a^{th} receive antenna of terminal j and the b^{th} transmit antenna of the BS on RB n. The maximum number of orthogonal spatial subchannels that can be used per RB in the considered cell is $\min(J\,M_R, M_T)$. The Figure 1 is a scenario representation in which there are two terminals with two receiver antennas each and 4 transmission antennas in the BS.

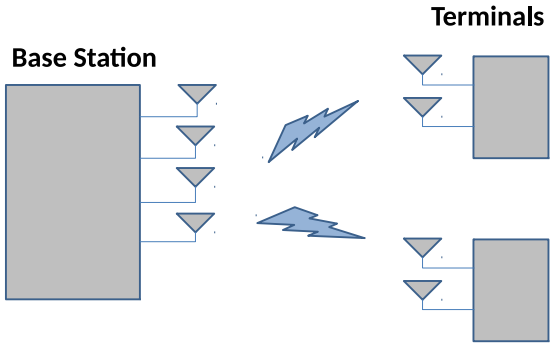


Figure 1. Transmission between a BS with 4 antennas and two terminals with 2 antennas.

We define SDMA group as a set of terminals spatially multiplexed in a given RB. Assuming, for example, three terminals and $M_T = M_R = 2$, the possible SDMA groups that can be built are $\{1\}$, $\{2\}$, $\{3\}$, $\{1,2\}$, $\{1,3\}$, $\{2,3\}$. In this sense, $\mathcal{G}$ consists in a set with the indices of all SDMA groups that can be built. The group $\{1,2,3\}$ cannot be considered because it has more terminals than antennas in the BS. In the previous example, the cardinality of $\mathcal{G}$ is $G = 6$. Assume that $\mathcal{G}_n$ is the set with the terminal's indices of the selected SDMA group assigned to RB n. The symbols to be transmitted

to terminal $j \in \mathcal{G}_n$ are filtered by a transmit matrix $\mathbf{M}_{j,n}$ whereas at the receiver they are filtered by a receive matrix $\mathbf{D}_{j,n}$. Note that $c_{j,n}$ is the number of streams transmitted to terminal j on RB n where $c_{j,n} \leq \min(M_T, M_R, \nu_{j,n})$, whereas $\nu_{j,n}$ is the rank of the channel matrix $\mathbf{H}_{j,n}$. Therefore, the input-output relation for the MIMO channel is given by

$$\widehat{\mathbf{b}}_{j,n} = \mathbf{D}_{j,n}\widetilde{\mathbf{b}}_{j,n} = \alpha_j \mathbf{D}_{j,n}\mathbf{H}_{j,n}\mathbf{M}_{j,n}\mathbf{b}_{j,n} + \alpha_j \mathbf{D}_{j,n}\mathbf{H}_{j,n} \sum_{i \in \mathcal{G}_n, i \neq j} (\mathbf{M}_{i,n}\mathbf{b}_{i,n}) + \mathbf{D}_{j,n}\mathbf{n}_{j,n}, \quad (1)$$

where $\widetilde{\mathbf{b}}_{j,n}$ and $\widehat{\mathbf{b}}_{j,n}$ are the prior-filtering and the post-filtering received signal vector of terminal j on RB n, respectively, $\mathbf{b}_{j,n}$ is the transmit signal vector of terminal j on RB n, $\mathbf{n}_{j,n}$ is the white Zero Mean Circularly Symmetric Complex Gaussian (ZMCSCG) noise vector of terminal j on RB n, and α_j represents the joint effect of the path loss and shadowing.

The Signal to Interference plus Noise Ratio (SINR) of the l^{th} stream of terminal j on RB n, $\gamma_{j,n,l}$, is given by

$$\gamma_{j,n,l} = \frac{\| \sqrt{\alpha_j} \mathbf{d}_{j,n}^l \mathbf{H}_{j,n}\mathbf{M}_{j,n}\mathbf{b}_{j,n} \|_2^2}{\| \sqrt{\alpha_j} \mathbf{d}_{j,n}^l \mathbf{H}_{j,n} \sum_{i \in \mathcal{G}_n, i \neq j} (\mathbf{M}_{i,n}\mathbf{b}_{i,n}) + \mathbf{d}_{j,n}^l \mathbf{n}_{j,n} \|_2^2}, \quad (2)$$

where $\mathbf{d}_{j,n}^l$ is the l^{th} row of matrix $\mathbf{D}_{j,n}$ and $\| \cdot \|_2$ denotes the 2-norm of a vector. We assume that the transmit power is fixed and equally distributed among the spatial subchannels.

Consider that the BS employs a link adaptation functionality that selects the best Modulation and Coding Scheme (MCS). Note that the choice of the best MCS depends on the channel transfer function as well as on the prior knowledge of the transmit and receive filters. Assuming that the chosen SDMA group for RB n, $\mathcal{G}_n$, corresponds to the g^{th} SDMA group in $\mathcal{G}$, the total transmit data rate of terminal j that belongs to the $g^{\text{th}} \in \mathcal{G}$ SDMA group is given by

$$r_{g,j,n} = \sum_{\forall l \in \mathcal{L}_{j,n}} f(\gamma_{j,n,l}), \quad (3)$$

where $f(\cdot)$ represents the link adaptation function and $\mathcal{L}_{j,n}$ is the set of spatial subchannels of terminal j on RB n.

We define $\mathbf{X}$ as an assignment matrix with binary-valued elements $x_{g,n}$ that assume 1 if SDMA group $g \in \mathcal{G}$ is assigned to RB $n \in \mathcal{N}$, and 0 otherwise. Let $\mathbf{O}$ be a binary matrix with elements $o_{g,j}$ that assume 1 if terminal j belongs to SDMA group g, and 0 otherwise. We assume that at the current TTI, terminal j has a data rate requirement equal to t_j, that a minimum of k_s terminals should be satisfied for service s and that the indices of the terminals in $r_{g,j,n}$, $o_{g,j}$ and in t_j are sequentially disposed according to the service i.e., the flows from j = 1 to $j = J_1$ are from service 1, flows from $j = J_1 + 1$ to $j = J_1 + J_2$ are from service 2, and so on.

2.2. BD-ZF

BD-ZF is a spatial filtering scheme whose main idea is to provide transmit and receive filters that enable to simultaneously transmit to multiple terminals in the same frequency resource (MU MIMO) without multi-user interference [Spencer et al. 2004]. It is a generalization of ZF precoding scheme for multi-antenna terminals in which multi-user interference is eliminated by projecting the signal of each terminal onto the kernel of the joint null space of the channels of all other terminals sharing the same RB. For more details, the interested reader can see [Spencer et al. 2004, Lima et al. 2014].

The dominant operation in the BD-ZF algorithm is the Singular Value Decomposition (SVD) performed on each terminal's channel matrix. According to [Björck 1996], the worst-case computational complexity when the SVD decomposition is applied in a matrix with dimension $M_R \times M_T$ is $\mathcal{O}\left(2M_R M_T^2 + 4M_T^3\right)$. Assuming that the SDMA group has J' terminals, the SVD is applied on the J' channel matrices and, therefore, the worst-case computational complexity is $\mathcal{O}\left(J'2M_R M_T^2 + J'4M_T^3\right)$.

2.3. URM problem and solution

According to the definitions presented before the URM problem can be formulated as follows [Lima et al. 2014]:

$$\max_{\mathbf{X}} \left(\sum_{g \in \mathcal{G}} \sum_{n \in \mathcal{N}} \sum_{j \in \mathcal{J}} x_{g,n}\, o_{g,j}\, r_{g,j,n} \right), \tag{4a}$$

subject to

$$\sum_{g \in \mathcal{G}} x_{g,n} = 1, \quad \forall n \in \mathcal{N}, \tag{4b}$$

$$x_{g,n} \in \{0,1\}, \quad \forall g \in \mathcal{G} \text{ and } \forall n \in \mathcal{N}. \tag{4c}$$

The objective function shown in (4a) is the total downlink data rate transmitted by the BS. The two constraints (4b) and (4c) assure that an RB will not be shared by different SDMA groups. The optimal solution of URM problem can be found according to the following steps:

1. Calculate $r_{g,j,n}$. In order to do that, we need to find the transmit and receive filters according to BD-ZF of all terminals for all possible SDMA groups and RBs;
2. For each RB, independently select the SDMA group with maximum total data rate, i.e., the SDMA group whose sum of the terminals' data rate is maximum.

Assuming that the estimated data rates $r_{g,j,n}$ are already calculated, i.e., the BD-ZF solution was already applied, the dominant operation of the optimal solution of URM problem is the computation of the total data rate of each SDMA group on each RB. According to the section 2.1, the maximum number of terminals in an SDMA group is limited by M_T. Therefore, the worst-case computational complexity of the optimal solution of the URM problem is $\mathcal{O}\left(GNM_T\right)$.

2.4. CRM problem and solution

According to [Lima et al. 2014], the CRM problem formulation is the same as the URM presented in (4) with the additional constraint that follows:

$$\sum_{j \in \mathcal{J}_s} u\left(\sum_{g \in \mathcal{G}} \sum_{n \in \mathcal{N}} x_{g,n}\, o_{g,j}\, r_{g,j,n}, t_j \right) \geq k_s, \ \forall s \in \mathcal{S}, \tag{5}$$

where $u(x, \beta)$ is a step function at β that assumes the value 1 if $x \geq \beta$ and 0 otherwise. Constraint (5) states that a minimum number of terminals should be satisfied for each service.

The optimal solution of CRM problem was described in [Lima et al. 2014] where it was formulated as ILP that can be optimally solved by BB-based algorithms [Nemhauser and Wosley 1999]. Note that, as in the URM solution, it is necessary to calculate $r_{g,j,n}$ to solve the CRM problem. Therefore, we need to find the transmit and receive filters according to BD-ZF of all terminals for all possible SDMA groups and RBs. Assuming that the estimated data rates $r_{g,j,n}$ are already calculated, the worst-case complexity of the optimal solution of the CRM problem is $\mathcal{O}\left(2^{GN}\right)$ [Lima et al. 2014].

3. SDMA Group Selection Framework

Our main idea is to reduce the number of SDMA groups used in the solutions. In other words, we intend to select $p\,G$ SDMA groups to be used to calculate the optimal solutions of URM and CRM problems where $0 < p < 1$.

Figure 2 shows a flowchart of the proposed framework. In step (1) we should identify all possible SDMA groups that can be composed. Then in step (2) we calculate the estimated Signal-to-Noise Ratio (SNR) of each link BS-terminal on all RBs as follows: $\bar{\gamma}_{j,n} = \dfrac{P.|\bar{h}_{j,n}|^2}{N_o}$, where P is the transmit power per spatial subchannel (fixed), N_o is the average noise power in the bandwidth of an RB and $\bar{h}_{j,n}$ consists in the mean of the elements of matrix $\mathbf{H}_{j,n}$. The average SNR is used in step (3) where we calculate a mean data rate, $\widehat{r}_{j,n}$, of terminal j on RB n. This mean data rate is obtained by the link adaptation lookup tables that map SNR regions to MCSs. The calculated data rate for each terminal j on each RB n is used in step (4) to calculate the SDMA grouping metric (see later). Then, in step (5) we select the $p\,G$ SDMA groups on each RB n with higher values for the considered metric. The remaining steps are present in the original solutions of URM and CRM problems in [Lima et al. 2014]. In step (6), the spatial filters are calculated for the selected SDMA groups in step (5). In step (7), the spatial filters are used to calculate the transmit data rate of each terminal j when belonging to the SDMA group g on RB n, $r_{g,j,n}$. Those data rates are input to the solutions of problems URM and CRM in step (8).

In this work we propose and evaluate different metrics to be used in step (4). Assume hereafter that $\widetilde{\mathcal{J}}_g$ consists in the set of terminals that belongs to the g^{th} SDMA group. The considered metric for the g^{th} SDMA group on RB n, $m_{g,n}$, is presented in Table 1.

The proposed metrics explore the mean data rates considering the SNR and number of terminals, which are variables that cause impact in the performance of group. The

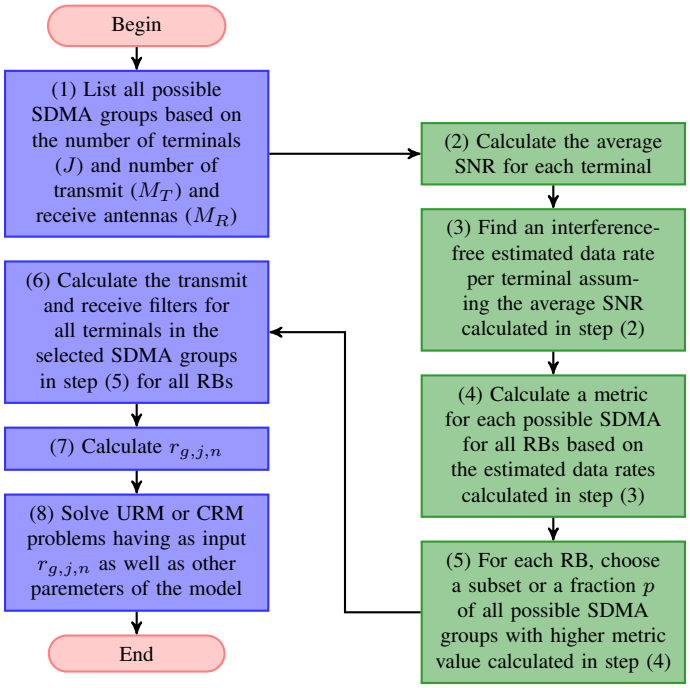


Figure 2. Flowchart of the proposed framework for solution of URM and CRM problems.

Table 1. Metrics for SDMA group selection.

Name	Equation	Comments
MEAN RATE	$m_{g,n} = \sum_{j \in \tilde{\mathcal{J}}_g} \widehat{r}_{j,n}$	Priority to SDMA groups with higher total transmit data rate
MAX MIN RATE	$m_{g,n} = \min_{j \in \tilde{\mathcal{J}}_g} (\widehat{r}_{j,n})$	Priority to SDMA groups with balanced terminal data rates or with a fair data rate distribution
SMALL GROUP MEAN RATE [1]	$m_{g,n} = \frac{\sum_{j \in \tilde{\mathcal{J}}_g} \widehat{r}_{j,n}}{\lvert\tilde{\mathcal{J}}_g\rvert}$	Variant of MEAN RATE that prioritizes the SDMA groups with small number of terminals
SMALL GROUP MAX MIN RATE	$m_{g,n} = \frac{\min_{j \in \tilde{\mathcal{J}}_g} (\widehat{r}_{j,n})}{\lvert\tilde{\mathcal{J}}_g\rvert}$	Variant of MAX MIN RATE that prioritizes the SDMA groups with small number of terminals

mean rate translate the channel quality. The number of terminal measures the spatial interference which degrades the channel quality.

For comparison purposes, we also consider a random scheme (RANDOM) where the choice of the $p\,G$ SDMA groups is done at random. With exception of the RANDOM strategy, the dominant operation in the computational complexity of the proposed selection metrics is the calculation of the average channel quality that requires $\mathcal{O}(M_R M_T)$. This operation is repeated for each terminal within the SDMA groups. As the number of terminals in an SDMA group is limited by M_T and the metric should be calculated for all SDMA groups on each RB, the worst-case computational complexity is $\mathcal{O}(GNM_T^2 M_R)$. Note that in some systems, an average wideband channel quality is already available and therefore, the complexity of this step is reduced to $\mathcal{O}(GNM_T)$.

4. Performance Results

[1] $|\cdot|$ represents the cardinality of a set.

Table 2. Main simulation parameters considered in the performance evaluation.

Parameter	Value	Unit
Cell radius	334	m
Transmit power per RB	0.8	W
Number of subcarriers per RBs	12	-
Number of RBs	10	-
Shadowing standard deviation	8	dB
Path loss [2]	$35.3 + 37.6 \cdot \log_{10}(d)$	dB
Noise spectral density	$3.16 \cdot 10^{-20}$	W/Hz
Number of snapshots	3000	-
Antenna configurations $M_R \times M_T$	2×2, 4×4 and 6×6	-
MIMO channel model	Classical IID	-
Number of services	2	-
Number of terminals per service	3	-
Required minimum number of satisfied terminals per service	2	- -

4.1. Simulation parameters

In this section we present the main assumptions that were considered for the performance evaluation of the proposed framework by means of computational simulations. We assume that the terminals are uniformly distributed in the cell coverage area. The links between BS and terminals are impaired by distance dependent path loss, lognormal shadowing and Independent and Identically Distributed (IID) Rayleigh-distributed fast fading. An RB consists of 12 subcarriers and 14 consecutive OFDM symbols [Lima et al. 2014]. We assume perfect Channel State Information (CSI) at the transmitter and receiver. We highlight here that CSI estimation is out of the scope of this work and the algorithms and methods proposed here do not rely on these aspects. Naturally, we expect that the absolute performance of the studied algorithms could be degraded when imperfect CSI estimation is present.

The simulations are organized by means of snapshots, where in a snapshot an RRA solution is provided. In every new snapshot, new independent samples of random variables are generated for channel state and other parameters of the model. The number of snapshots was chosen so as to assure statistical confidence of the presented results. We assume that the link adaptation is performed based on the report of 15 discrete Channel Quality Indicators (CQIs) used by the LTE system [Lima et al. 2014]. The Signal to Noise Ratios (SNRs) thresholds for MCS switching used here were the same ones employed in [Lima et al. 2014]. The main simulation parameters are shown in Table 2.

The ILP problems were solved using IBM ILOG CPLEX Optimizer [IBM 2009]. In the plots, we call "Original solution" the optimal solution of the CRM or URM problems that considers all possible SDMA groups. When the performance metrics are concerned, we consider two main ones: outage rate and total data rate. An outage event happens when an algorithm cannot manage to find a feasible solution for CRM problem, i.e., the algorithm does not find a solution fulfilling the constraints of CRM problem. Outage rate is defined as the ratio between the number of snapshots with outage events and the total number of simulated snapshots. Therefore, this performance metric shows the capability of the algorithms in finding a feasible solution to CRM problem. The outage

[2] d is the distance between the base station and the terminal in meters.

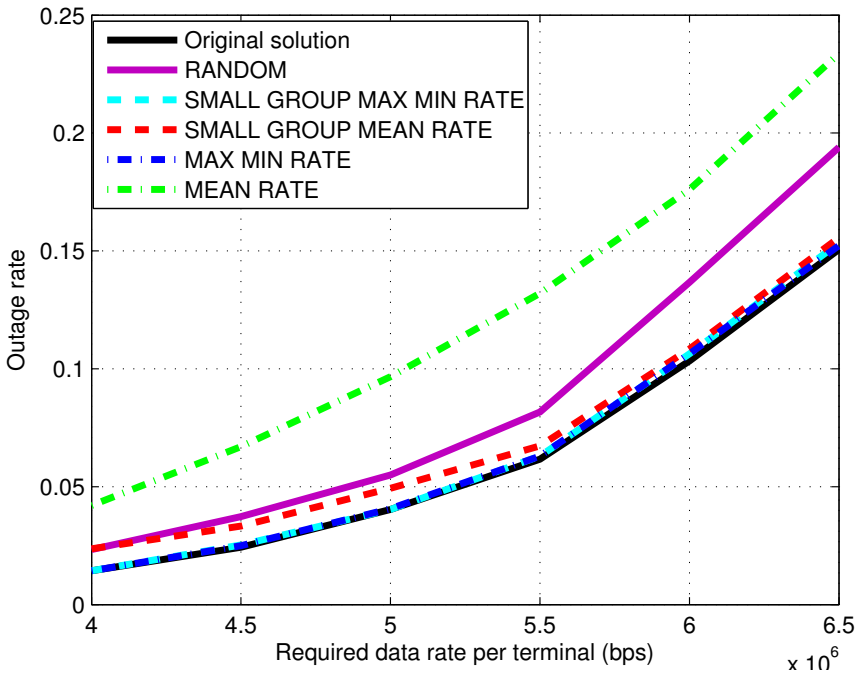


Figure 3. Outage rate versus required data rate with the optimal solution of the CRM problem with $p = 1$ (all SDMA groups are selected) and $p = 0.6$ for different metrics.

rate is meaningless for URM problem. The total data rate is the sum of the data rates obtained by all the terminals in the cell in a given snapshot. This metric is suitable for both URM and CRM problems. Finally, increments in the offered load are emulated by increasing the data rate requirements of the terminals.

4.2. Simulation Results

In Figures 3 and 4 we can see the outage rate versus the required data rate per terminal for the proposed framework with different proportion of selected groups: 60% and 30%, respectively. Let p the groups percentage considered in the problem. The objective is to compare the performance of the original solution of CRM problem where all SDMA groups are considered ($p = 1$), and the solutions using our proposed framework with different values of p with the metrics presented in section 3. As expected, the solution with $p = 1$ presents the best performance in terms of outage since it has more options of SDMA groups per RB in order to find an RRA solution that satisfies the problem constraints.

The worst performance in Figures 3 and 4 is achieved by the MEAN RATE metric. This metric selects the SDMA groups with higher mean data rate. Therefore, it is expected that most of the selected SDMA groups will be composed of the terminals with better channel qualities. Consequently, the RRA solution does not have many options to find feasible associations between RBs and SDMA groups that contain the other terminals in medium and poor channel states leading to a higher outage rate. The second worst performance is achieved with the RANDOM metric. Basically, this metric does not consider any information of terminals or channel states in order to select the SDMA groups.

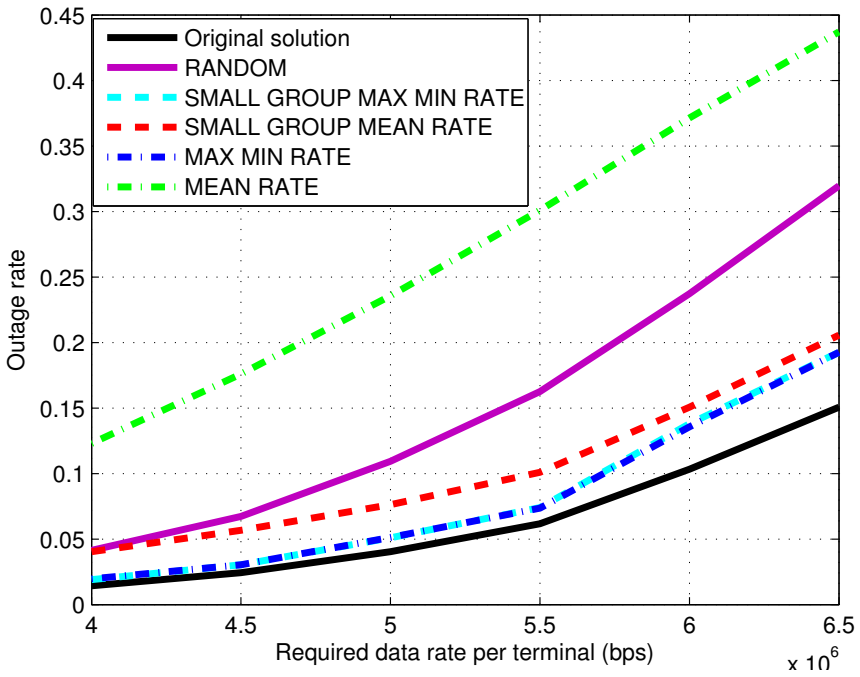


Figure 4. Outage rate versus required data rate with the optimal solution of the CRM problem with $p = 1$ (all SDMA groups are selected) and $p = 0.3$ for different metrics.

In both figures we can see that the SMALL GROUP MEAN RATE metric presents better performance than MEAN RATE metric. This metric SMALL GROUP MEAN RATE gives more opportunities to select terminals in medium and poor channel states since the data rate is not the unique factor to be considered. Focusing on Figure 3 we can see that the metrics MAX MIN RATE and SMALL GROUP MAX MIN RATE perform almost optimally. This means that the proposed framework achieves a practically optimal solution selecting only 60% of the SDMA groups. In Figure 4, we can observe again that the MAX MIN RATE and SMALL GROUP MAX MIN RATE present the best performance. By choosing the SDMA groups with maximum worst terminal data rate, those metrics are capable of providing a small and acceptable degradation in performance even when only 30% of the SDMA groups are considered. In particular, the SMALL GROUP MAX MIN RATE has the advantage of selecting SDMA groups with small size and therefore, the computational complexity of spatial filter calculation is reduced. It is worth of mentioning that all metrics present reduced performance loss compared to the solution with $p = 1$ as we increase the value of p. This can be seen by comparing Figures 3 and 4.

In Figure 5 we present the Cumulative Distribution Function (CDF) of the total data rate for the same scenario of Figure 4 when the required data rate per terminal is 6 Mbps. The samples used in the CDFs were taken from the snapshots in which all solutions managed to find a feasible solution, i.e., there is no outage. We can see that the solution with $p = 1$ presents the best performance, as expected. The poor performance of the RANDOM metric is due its unawareness regarding channel state information as explained before. All other metrics presents a similar performance with a loss in the 50^{th}

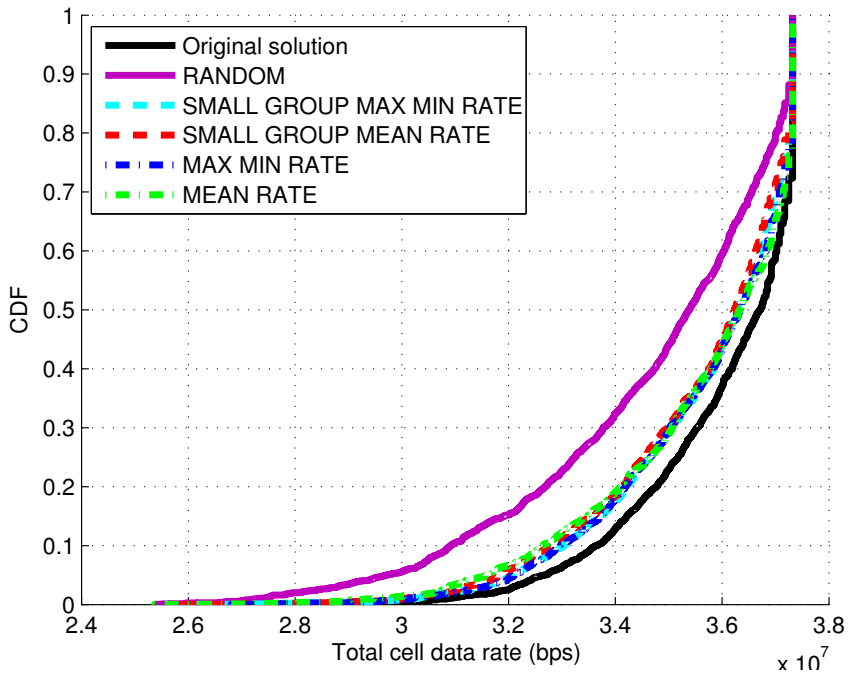


Figure 5. CDF of total data rate for the data rate requirement of 6 Mbps with the optimal solution of the CRM problem with $p = 1$ (all SDMA groups are selected) and $p = 0.3$ for different metrics.

percentile of only 1% compared to the original solution. Although it is not shown here, the performance loss of the proposed metrics are decreased as the terminals' required data rate is reduced.

In Figure 6 we present the mean of the total data rate for URM problem considering $p = 0.3$. As we can see in this figure, selecting only 30% of the SDMA groups for solution does not lead to significant performance losses compared to the original solution. Therefore, we will have an acceptable loss for a large reduction of groups. The exception is the RANDOM metric that presents a loss of 1.5 Mbps in the mean of the total data rate compared to the solution with $p = 1$, a reduction only 4%. The others metrics don't have loss greater than 0.7 Mbps. This performance is achieve because the problem don't have QoS restrictions. Users with good channel state will receive many RB, making easy increase the global rate.

In the following we provide some comments about the computational complexity of the studied solutions. When a fraction p of G possible groups is selected, we need to calculate the spatial filters of the terminals of only pG SDMA groups in the proposed framework for each RB. On the other hand, the original solution should calculate the spatial filters of the terminals of all G possible SDMA groups for each RB. According to the computational complexity of the BD-ZF presented in section 2.2, the proposed framework avoids the calculation of approximately $NG(1-p)M_R M_T^3 + NG(1-p)M_T^4$ operations compared to the original solution assuming that the maximum number of terminals in an SDMA group is M_T. Note that the complexity to calculate the metrics of all SDMA groups is a polynomial with order lower than the calculation of all the spatial filters with

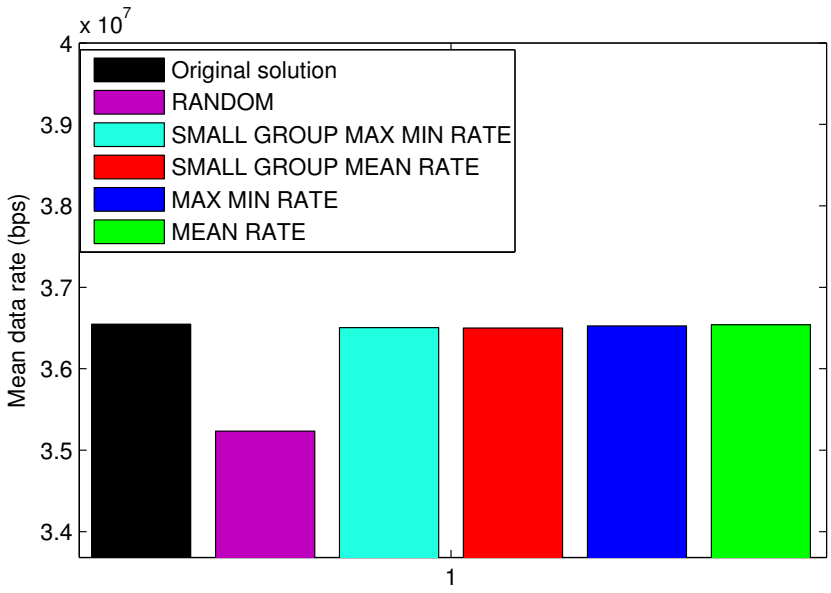


Figure 6. MEAN of total data rate with the optimal solution of the URM problem with $p = 1$ (all SDMA groups are selected) and $p = 0.3$ for different metrics.

BD-ZF as shown in section 3.

Besides, the reduction in computational complexity is also present in the optimal CRM and URM solutions. As it was shown in section 2.4, the complexity of the original solution of CRM problem is $\mathcal{O}\left(2^{GN}\right)$ and with the proposed framework is $\mathcal{O}\left(2^{pGN}\right)$. The worst-case complexity of the original solution of URM problem is $\mathcal{O}\left(GNM_T\right)$ and with proposed framework is $\mathcal{O}\left(pGNM_T\right)$.

The presented results in this section have shown that significant computational work can be saved by selecting only a fraction of the possible SDMA groups when solving the CRM and URM problems. This selection should be done with the help of some of the metrics proposed in this work. In particular, the SMALL GROUP MAX MIN RATE has shown to be effective for both problems.

5. Conclusions and Perspectives

Although Radio Resource Allocation (RRA) algorithms applied in Multiple Input Multiple Output (MIMO) systems can lead to important performance gains, they are in general very complex and impose a high computational burden. Therefore, strategies that decrease the computational complexity with low performance loss are welcome. In this work we revisited the Constrained Rate Maximization (CRM) and Unconstrained Rate Maximization (URM) problems. In order to obtain the optimal solution of those problems, it is needed to list all possible Space Division Multiple Access (SDMA) groups and calculate the transmit and receive filters of the terminals. Our proposed was to select only a fraction of all the possible SDMA groups based on specific metrics based on Channel State Information (CSI) in order to solve CRM and URM problems. Accordingly, there is an overall computational complexity reduction compared to the original solution.

By computer simulations, we have shown that some of the proposed selection

metrics are capable of providing a small performance degradation compared to the original solution even when only 30% of the possible SDMA groups are selected. Therefore, the proposed framework is capable of achieving a good performance-computational complexity trade off. As the proposed framework is a flexible and general, we point out as perspectives the study of the proposed framework on other RRA problems for MIMO systems.

Acknowledgment

This work was supported by the Innovation Center Ericsson Telecomunicações S.A., Brazil, under EDB/UFC.30 Technical Cooperation Contract. The student Lászlon R. Costa would like to thank the FUNCAP for his financial support.

References

Alamouti, S. (1998). A Simple Transmit Diversity Technique for Wireless Communications. *IEEE Journal on Selected Areas in Communications*, 16(8):1451–1458.

Björck, A. (1996). *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics.

Caire, G. and Shamai, S. (2003). On the Achievable Throughput of a Multiantenna Gaussian Broadcast Channel. *IEEE Transactions on Information Theory*, 49(7):1691 – 1706.

Costa, M. (1983). Writing on Dirty Paper. *IEEE Transactions on Information Theory*, 29(3):439 – 441.

Foschini, G. J. (1996). Layered Space-Time Architecture for Wireless Communication in a Fading Environment when Using Multi-Element Antennas. *Bell Labs Technical Journal*, 1(2):41–59.

Gesbert, D., Kountouris, M., Heath, R. W., Chae, C.-B., and Salzer, T. (2007). Shifting the MIMO Paradigm. *IEEE Signal Processing Magazine*, 24(5):36 –46.

Ho, W. W. L. and Liang, Y.-C. (2009). Optimal Resource Allocation for Multiuser MIMO-OFDM Systems with User Rate Constraints. *IEEE Transactions on Vehicular Technology*, 58(3):1190–1203.

IBM (2009). IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/. Accessed: 12-09-2015.

Li, Q., Li, G., Lee, W., Lee, M., Mazzarese, D., Clerckx, B., and Li, Z. (2010). MIMO Techniques in WiMAX and LTE: a Feature Overview. 48(5):86–92.

Lima, F. R. M., Maciel, T. F., Freitas, W. C., and Cavalcanti, F. R. P. (2014). Improved Spectral Efficiency with Acceptable Service Provision in Multi-User MIMO Scenarios. *IEEE Transactions on Vehicular Technology*, 63(6):2697–2711.

Mietzner, J., Schober, R., Lampe, L., Gerstacker, W. H., and Hoeher, P. A. (2009). Multiple-Aantenna Techniques for Wireless Communications - A Comprehensive Literature Survey. *IEEE Communications Surveys Tutorials*, 11(2):87–105.

Nemhauser, G. and Wosley, L. (1999). *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1nd edition.

Spencer, Q. H., Swindlehurst, A. L., and Haardt, M. (2004). Zero-Forcing Methods for Downlink Spatial Multiplexing in Multiuser MIMO Channels. *IEEE Transactions on Signal Processing*, 52(2):461 – 471.

Tejera, P., Utschick, W., Bauch, G., and Nossek, J. A. (2006). Subchannel Allocation in Multiuser Multiple-Input Multiple-Output Systems. *IEEE Transactions on Information Theory*, 52(10):4721 –4733.

Análise e solução para o problema da instabilidade de associação em redes IEEE 802.11 densas

Helga Balbi[1], Diego Passos[1], Ricardo Carrano[2], Luiz Magalhaes[2], Célio Albuquerque[1]

[1]Instituto de Computação
– Universidade Federal Fluminense (UFF) – Niterói – RJ – Brazil

[2]Departamento de Engenharia de Telecomunicações
– Universidade Federal Fluminense (UFF) – Niterói – RJ – Brazil

{helgadb,diego,carrano,schara,celio}@midiacom.uff.br

Abstract. *The decision of whether or not to perform a handoff between access points in an infrastructured IEEE 802.11 network is taken exclusively by the stations. Even without mobility, client devices may decide to migrate to another access point with the goal of improving performance. However, the criteria used to perform handoffs are not defined by the IEEE 802.11 standard and, thus, are dependent on specific vendor implementations. In this paper, we use data from a real large scale production network deployed at Universidade Federal Fluminense (UFF) and run experiments to demonstrate that such implementations are commonly deficient, resulting in high levels of association instability in dense environments. By analyzing the implementation used by the most common devices found on the Wi-Fi network at UFF, we were able to conclude that this instability, which we refer to as "ping-pong", results from the direct usage of RSSI measurements which are highly variable. Finally, we propose a simple process to smooth the time series of these samples. Experiments performed in a real network show that this smoothing effectively eliminates the occurrence of "ping-pongs".*

Resumo. *A decisão de realização ou não de handoff entre pontos de acesso em uma rede IEEE 802.11 infraestruturada é uma prerrogativa exclusiva das estações. Mesmo sem mobilidade, dispositivos clientes podem decidir migrar para um novo ponto de acesso com o objetivo de obter melhor desempenho. Entretanto, os critérios utilizados para a realização do handoff não são definidos no padrão IEEE 802.11, ficando a cargo das implementações dos fabricantes. Neste trabalho, realizando-se experimentos e utilizando-se dados reais de uma rede sem fio de larga escala implantada na Universidade Federal Fluminense (UFF), demonstra-se que tais implementações são comumente deficientes, resultando em grande instabilidade de associação em ambientes densos. Através da análise da implementação mais comumente utilizada por clientes na rede Wi-Fi da UFF, concluiu-se que, esta instabilidade, denominada neste trabalho de "ping-pong", resulta da utilização direta de amostras de RSSI que são altamente variáveis. Finalmente, propõe-se um processo simples de suavização da série temporal destas amostras. Experimentos realizados em uma rede real mostram que esta suavização efetivamente elimina o "ping-pong".*

1. Introdução

Atualmente, a tecnologia IEEE 802.11 [IEEE 2012] é amplamente empregada em ambientes residenciais, corporativos e de eventos, oferecendo conectividade de rede sem fio a dispositivos móveis. No decorrer dos anos, com a redução dos custos relacionados a esta tecnologia, redes IEEE 802.11 infraestruturadas densas, compostas por milhares de estações cliente (STAs) e centenas de pontos de acesso (APs) instalados para prover cobertura e capacidade aos usuários, se tornaram comuns [Magalhães et al. 2013]. Muitas vezes, em um mesmo ambiente, uma STA pode estar ao alcance de diversos APs que compõem o *Extended Service Set* (ESS), tendo a possibilidade de se conectar a qualquer um deles [Vasudevan et al. 2005]. A necessidade de redes densas se torna ainda maior em certos contextos, como o da Internet das Coisas, no qual interfaces de comunicação residirão em qualquer objeto utilizado no dia a dia das pessoas, possibilitando o desenvolvimento de novas aplicações nas áreas de transporte, logística, saúde, segurança e conforto [Tan e Wang 2010].

O padrão IEEE 802.11 define que a STA deve escolher a qual AP irá se conectar, mas não define como esta escolha deve ser realizada. A implementação do algoritmo de escolha varia para diferentes fabricantes e, comumente, é baseada no RSSI (*Received Signal Strength Indication*), que é um indicador do nível de sinal recebido do AP. Em geral, o AP que apresenta maior RSSI é escolhido em detrimento dos outros. Outra função comumente realizada pela STA, e que, conforme será mostrado neste trabalho, também costuma depender do RSSI, é a migração espontânea entre APs de um mesmo ESS em busca de melhor desempenho, processo denominado comumente por *handoff*[1]. Assim como a escolha inicial de um AP, os critérios para migração não são definidos pelo padrão, ficando a cargo da implementação dos fabricantes [Vasudevan et al. 2005].

Como será demonstrado neste artigo, estas implementações são comumente ineficientes, resultando em trocas de AP frequentes, o que pode ser inconveniente para o usuário. Idealmente, uma migração para um novo AP deveria ocorrer apenas quando, de fato, há um ganho de desempenho. Um exemplo é um usuário móvel que gradativamente se afasta do seu AP atual, aproximando-se de outro AP da rede. Entretanto, especialmente em ambientes densos, verifica-se que muitas vezes a migração leva uma STA estática a se associar a outro AP de desempenho muito similar ou até inferior ao atual. Mais grave, em certos casos uma STA realizará migrações sucessivas, alternando várias vezes em um curto período entre dois ou mais APs. Este comportamento, que foi previamente citado em [Raghavendra et al. 2007], causa prejuízo aos usuários por resultar em interrupções na conectividade e quebras de conexões ativas, além de ocasionar troca de quadros desnecessários na rede. Esta instabilidade é comumente referida como "ping-pong" na literatura [Mhatre e Papagiannaki 2006].

As migrações excessivas prejudicam, particularmente, aplicações interativas ao ocasionar interrupções na conexão. Além disso, o processo de migração requer a troca de mensagens de controle, o que acarreta um maior gasto energético, algo indesejado em dispositivos móveis. Os inconvenientes são mais graves nos casos em que a migração é mais custosa em termos de tempo, como em redes que necessitam de nova autenticação do cliente baseada em 802.1X, e nos casos em que uma nova autenticação do usuário via

[1]Neste trabalho, "*handoff*" e "migração entre APs" serão utilizados com o mesmo significado.

captive portal é necessária. Em certos casos, as migrações frequentes podem impossibilitar a utilização da rede [Raghavendra et al. 2007].

Embora o processo de migração entre APs já tenha sido estudado na literatura, como em [Mishra et al. 2003], [Ramani e Savage 2005] e [Huang et al. 2006], os trabalhos existentes, normalmente, têm foco na redução do tempo necessário para a realização da migração. Há ainda trabalhos que buscam otimizar as migrações de estações móveis com base no seu trajeto [Kim et al. 2012]. Entretanto, a questão do ping-pong, que afeta, inclusive, estações estáticas, comumente, não é analisada nas soluções propostas.

Neste trabalho, utilizando como estudo de caso a rede Wi-Fi de produção da Universidade Federal Fluminense (UFF), será mostrada a severidade do problema do ping-pong, que ocorre de forma recorrente para dispositivos de uma grande gama de fabricantes. Uma análise profunda e detalhada do código fonte que implementa o algoritmo de *handoff* utilizado pelos dispositivos mais populares entre os usuários da rede Wi-Fi UFF mostra que a raiz do problema está na utilização direta de medidas de RSSI, um valor que, conforme será mostrado, possui alta variabilidade no tempo. Com base nesta análise, uma solução que aplica a suavização destas variações é proposta. Por fim, o trabalho apresenta os resultados dos testes experimentais comparativos realizados em rede real, mostrando que o ping-pong foi efetivamente evitado com a utilização da solução proposta.

As próximas seções estão organizadas da seguinte forma: a Seção 2 apresentará a caracterização do problema do ping-pong e mostrará que o problema é comum e recorrente em redes densas, e, especificamente, na rede analisada como estudo de caso; a Seção 3 analisará as causas do problema do ping-pong, tendo como base os dispositivos mais populares na rede, e proporá uma solução baseada na suavização dos valores de RSSI; a Seção 4 mostrará os resultados dos testes comparativos para avaliação da solução proposta; a Seção 5 tratará dos trabalhos relacionados ao tema e discutirá os compromissos da solução proposta; e, por fim, a Seção 6 apresentará as conclusões do trabalho e propostas para trabalhos futuros.

2. Caracterização do problema e estudo de caso

Esta seção apresentará a análise da ocorrência do problema do ping-pong em uma rede real em produção, atualmente composta por 390 pontos de acesso instalados em diversos *campi* da UFF, em Niterói, RJ [Magalhães et al. 2013]. Os pontos de acesso utilizados na rede Wi-Fi UFF rodam o Linux embarcado OpenWrt e o *daemon hostapd*, responsável por realizar funções de um AP do padrão IEEE 802.11, como o envio de *beacons*, associação/desassociação e autenticação/desautenticação das estações cliente. Este *daemon* gera *logs* de cada um dos eventos de autenticação/desautenticação e associação/desassociação, que, por sua vez, são enviados para um servidor centralizado na rede. Através dos *logs*, é possível avaliar em qual momento um determinado cliente se conectou/desconectou a um determinado AP. Para este trabalho, foram coletados e analisados *logs* no período entre 0h do dia primeiro e 3h:45min do dia 26 de abril de 2015. A sequência de eventos no *log* esperados para um cliente que se conecta à rede em questão é: (1) autenticação do cliente; (2) associação do cliente; (3) autenticação WPA2 Enterprise do cliente; e (4) desassociação e/ou desautenticação do cliente. Neste trabalho, uma

Tabela 1. Descrição dos parâmetros utilizados para a caracterização do ping-pong.

Parâmetro	Descrição	Valor
$Xmax$	Intervalo de tempo máximo entre conexões a APs diferentes e consecutivos que caracteriza o ping-pong.	30 segundos
$Zmax$	Intervalo de tempo máximo entre a desconexão de um AP de origem e conexão a um AP de destino, que caracteriza o *handoff*.	2 segundos
$Nmin$	Número mínimo de migrações entre APs, que devem ocorrer consecutivamente respeitando os limiares de tempo estabelecidos, para que o ping-pong seja caracterizado.	2 migrações

conexão[2] de uma STA a um AP é definida se há nos *logs* ao menos um dos eventos de autenticação ou associação. De forma análoga, uma desconexão da STA é definida se há nos *logs* ao menos um dos eventos de desassociação ou desautenticação.

A Tabela 1 mostra a descrição dos parâmetros que foram utilizados para a caracterização do ping-pong. Para que o ping-pong ocorra, mais de um *handoff* deve ocorrer em um período determinado. Neste trabalho, o *handoff* é definido como a desconexão espontânea do cliente de um AP de origem e a subsequente conexão a um AP de destino (*e.g.*, se um cliente se desassocia de um AP e se associa novamente à rede no dia seguinte, isso não configura um *handoff*). A decisão do *handoff* varia de acordo com o fabricante do dispositivo cliente e de configurações do *software* utilizado para esta função. Estudos anteriores, como [Mishra et al. 2003], mostram que diferentes dispositivos podem apresentar diferentes comportamentos durante a realização do *handoff*, resultando em durações variáveis para este processo. Para lidar com esta variabilidade, a caracterização do problema do ping-pong utiliza o parâmetro $Zmax$, que define o intervalo de tempo máximo entre a desconexão de um AP anterior e a conexão a um novo AP caracterizando um *handoff*. Em outras palavras, se este intervalo for maior que $Zmax$, este processo não configura um *handoff* e, portanto, não pode ser parte de um ping-pong.

Além da ocorrência do *handoff*, para que o ping-pong seja caracterizado também é necessário que o cliente permaneça pouco tempo no AP de origem. A definição do que seria "pouco tempo" é subjetiva e deve levar em conta a degradação de qualidade do acesso percebida pelo usuário. Neste trabalho, este tempo é limitado por $Xmax$, que define o tempo máximo entre conexões consecutivas a diferentes APs caracterizando a ocorrência de ping-pong. Além dos limiares de tempo, um número N de migrações consecutivas entre APs mínimo, denominado $Nmin$, deve ocorrer respeitando estes limiares. Após o limiar $Nmin$ ser alcançado, as demais migrações que se enquadrarem nos limiares de tempo $Zmax$ e $Xmax$ ocasionarão o incremento do número de ping-pongs observados para o cliente. Caso uma migração que não se enquadre nos limiares ocorra, o contador N é zerado. É importante notar que o ping-pong pode envolver um número variável de APs, tendo em vista que, em um mesmo ambiente, um cliente poderá encontrar diversos APs pertencentes ao mesmo ESS e migrar entre eles. Desta forma, o número de APs diferentes

[2]Como nos *logs* analisados um evento de associação pode se referir tanto à associação inicial quanto à reassociação de uma STA a um AP, deste ponto em diante o termo "conexão" será utilizado para denotar ambos.

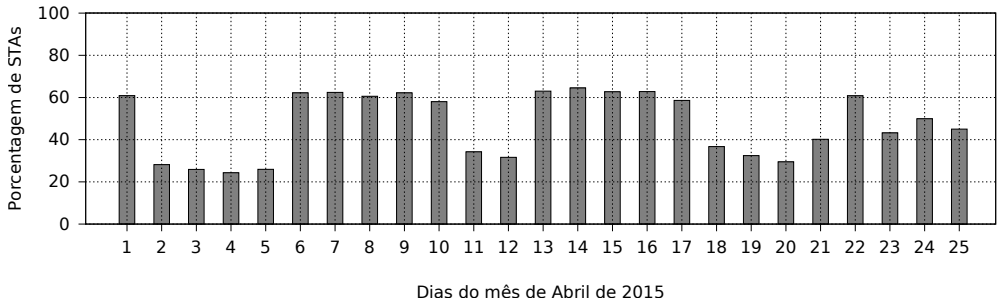


Figura 1. Porcentagem de STAs que sofreram ping-pong a cada dia analisado.

envolvidos nos eventos de ping-pongs não foi limitado. O único requisito é que os APs de origem e destino em uma migração sejam diferentes.

Nas análises apresentadas nesta seção, conforme mostra a Tabela 1, os valores utilizados para os parâmetros definidos foram: $Xmax = 30$ segundos; $Zmax = 2$ segundos e $Nmin = 2$ migrações. $Zmax$ foi escolhido com base em resultados anteriores, como o de [Mishra et al. 2003], e adicionada uma margem para enquadrar variações de tempo para realização do *handoff* que possam vir a ocorrer para diferentes dispositivos. Os valores de $Xmax$ e $Nmin$ foram escolhidos considerando-se que, se no máximo em 1 minuto o usuário realizou dois *handoffs*, ao menos um deles foi desnecessário. Com estes valores, é esperado que os casos em que o ping-pong gera incomodo ao usuário da rede sejam representados.

No decorrer das análises dos *logs*, um total de 15.783 STAs diferentes, com pelo menos um ciclo de conexão/desconexão e que tinham credenciais para acesso à rede, foram encontradas. A Figura 1 mostra, para cada dia do *log* analisado, a porcentagem destas STAs que sofreram ao menos um ping-pong. Os resultados mostram que, nos dias referentes a feriados, pontos facultativos e finais de semana, que são 2, 3, 4, 5, 11, 12, 18, 19, 20, 21, 23, 24 e 25, o ping-pong ocorreu com menor frequência. Já nos demais dias, o ping-pong foi observado para mais de 50% das STAs. O dia 26 foi excluído do gráfico porque apresentou dados somente até às 3:54 hs da tarde.

Um dos motivos que podem explicar o maior número de ping-pongs observados nos dias úteis é o aumento do tempo total de conexão observado para as STAs. Este aumento do tempo de permanência no ambiente aumenta a possibilidade de que o cliente experimente o ping-pong. No dia 4, por exemplo, no qual 24% das STAs sofreram ping-pong, a porcentagem de STAs que permaneceram mais de 10 minutos conectadas foi apenas $38{,}48\%$. Dentre estas, 54% sofreram ping-pongs. Já no dia 14, no qual $64{,}53\%$ das STAs sofreram ping-pong, a porcentagem de STAs que permaneceram mais de 10 minutos conectadas subiu para 86% e, destas, $72{,}19\%$ sofreram ping-pongs. Um comportamento semelhante pode ser observado para os demais dias. Outra questão que pode influenciar a ocorrência de ping-pongs é a movimentação das pessoas e objetos pelo ambiente analisado. Esta movimentação pode ocasionar varições momentâneas no RSSI amostrado pelas STAs, estimulando mais ocorrências de ping-pongs. Desta forma, a circulação de um maior número de pessoas estimula o aumento no número de ping-pongs observados nos dias úteis.

Dentre os dias úteis, o número total de ocorrências do ping-pong por dia variou

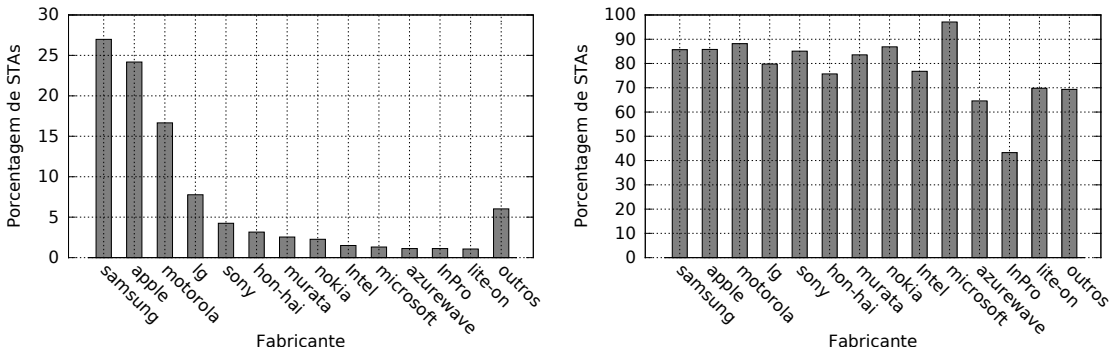


Figura 2. (a) Distribuição de dispositivos por fabricantes mais populares encontrados na rede; (b) Porcentagem dos dispositivos de cada fabricante que sofreram ping-pong.

entre 35.170 e 67.155. O dia com maior número de ocorrências foi o dia 17. Neste dia, a média do número de ping-pongs por STAs que sofreram o problema foi $17,83$. O usuário com maior número de ocorrências neste dia apresentou 1.007 ping-pongs e permaneceu na rede durante $7,4$ horas. No dia 10, no qual o número de ocorrências foi o menor dentre os demais, a média foi de $9,3$ ping-pongs por STAs que sofreram o problema. O usuário com maior número de ocorrências neste dia apresentou 683 ping-pongs e permaneceu $6,63$ horas conectado na rede.

As estações que sofreram ao menos um ping-pong estão suscetíveis aos inconvenientes que podem ser ocasionados pelo *handoff* indesejado em um ESS. Os resultados mostram que a ocorrência de ping-pongs é comum e recorrente na rede analisada.

A Figura 2 mostra estatísticas relativas aos fabricantes dos dispositivos encontrados. Nesta análise, para identificar o fabricante de cada STA, foi utilizada a base de dados fornecida por [Stiller 2015], baseada nos registros do próprio IEEE que relacionam o endereço MAC do dispositivo ao fabricante. Os gráficos apresentam os resultados para fabricantes com mais de 1% de ocorrências, dentre o total. A Figura 2(a) mostra a distribuição de STAs por fabricante. Os dispositivos dos fabricantes Samsung, Apple, Lenovo/Motorola e LG foram os mais encontrados, somando aproximadamente 76% do total de dispositivos. A Figura 2(b) mostra a porcentagem de STAs de cada fabricante que sofreram ao menos um ping-pong em relação a todas as STAs do respectivo fabricante. Os resultados mostram que a ocorrência de ping-pongs é comum entre diferentes fabricantes e ocorreu para mais de 70% das STAs dos principais fabricantes listados. No caso dos dispositivos Samsung, 85% das STAs sofreram ping-pongs.

Além dos *logs* gerados pelo *hostapd*, os *logs* gerados pelos servidores DHCP (*Dynamic Host Configuration Protocol*) da rede foram analisados para o mesmo período dentre os dias 1 e 26 de abril de 2015. A informação avaliada a partir destes *logs* foi relativa ao *hostname* requisitado pelas STAs. Tendo em vista que dispositivos móveis comumente apresentam *hostname* que segue determinado padrão, esta informação foi coletada para inferir o sistema operacional (SO) utilizado pelas STAs da rede e gerar estatísticas relativas a este parâmetro.

A Figura 3(a) mostra a distribuição de STAs de acordo com o seu sistema operaci-

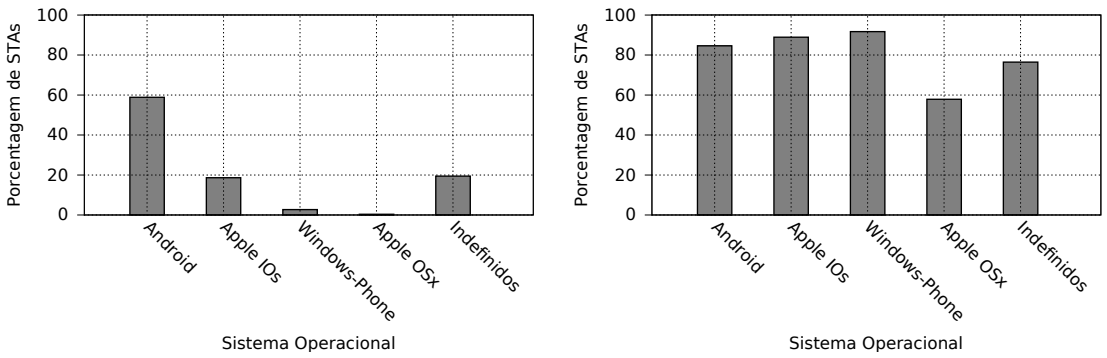


Figura 3. (a) Distribuição de dispositivos por sistema operacional; (b) porcentagem dos dispositivos de cada sistema operacional que sofreram ping-pong.

onal. Os resultados mostram que 60% das estações utilizam sistema operacional Android. A coluna "Indefinidos" refere-se às STAs cujo *hostname* não segue um padrão conhecido. Desta forma, para estas STAs, não foi possível estimar o sistema operacional utilizado. A Figura 3(b) mostra a porcentagem de STAs que sofreu ao menos um ping-pong para os diferentes sistemas operacionais no decorrer do período analisado. Os resultados mostram que aproximadamente 80% dos dispositivos Android, que são os mais encontrados na rede, sofreram ao menos um ping-pong.

3. Análise da origem do problema e proposta de solução

Como a decisão de realização do *handoff* é originada na STA, é intuitivo que a solução para o problema do ping-pong seja implementada nestes dispositivos. Um ponto de partida para uma proposta de solução é a avaliação do funcionamento dos dispositivos que operam com sistema Android, já que, conforme mostrado na Seção 2, representam 60% das STAs da rede utilizada como estudo de caso neste trabalho. Além disso, o código implementado neste sistema é aberto, e, portanto, mais acessível em relação aos demais sistemas, como os da Apple e Microsoft.

Assim como distribuições Linux, o Android comumente utiliza o *software Wpa Supplicant* [Linux Wireless Wiki 2015] para a realização do processo de autenticação do cliente na rede. Além da autenticação, o *Wpa Supplicant* pode ser configurado para disparar a realização de varreduras espectrais em busca de APs vizinhos, e *handoff* entre APs de um ESS. Em dispositivos móveis, esta configuração é comumente definida pelo fabricante, podendo, em alguns casos, estar desabilitada. Neste caso, o dispositivo não irá realizar *handoff* automaticamente. Porém, os resultados apresentados na Seção 2 mostraram que a realização de *handoffs* por parte das STAs é comum.

A Figura 4 (a) mostra o fluxograma de decisão do *handoff* do *Wpa Supplicant*. Para realizar o *handoff*, o *Wpa Supplicant* requisita informações de varredura para obter dados sobre APs nas redondezas. A varredura pode ser requisitada ao *driver* da interface sem fio pelo *Wpa Supplicant* ou por outros *softwares*. Quando o *Wpa Supplicant* realiza esta função, as varreduras ocorrem em intervalos que variam de acordo com a qualidade da conexão da STA ao AP. Desta forma, mais varreduras poderão ser realizadas caso a qualidade da comunicação se torne pior.

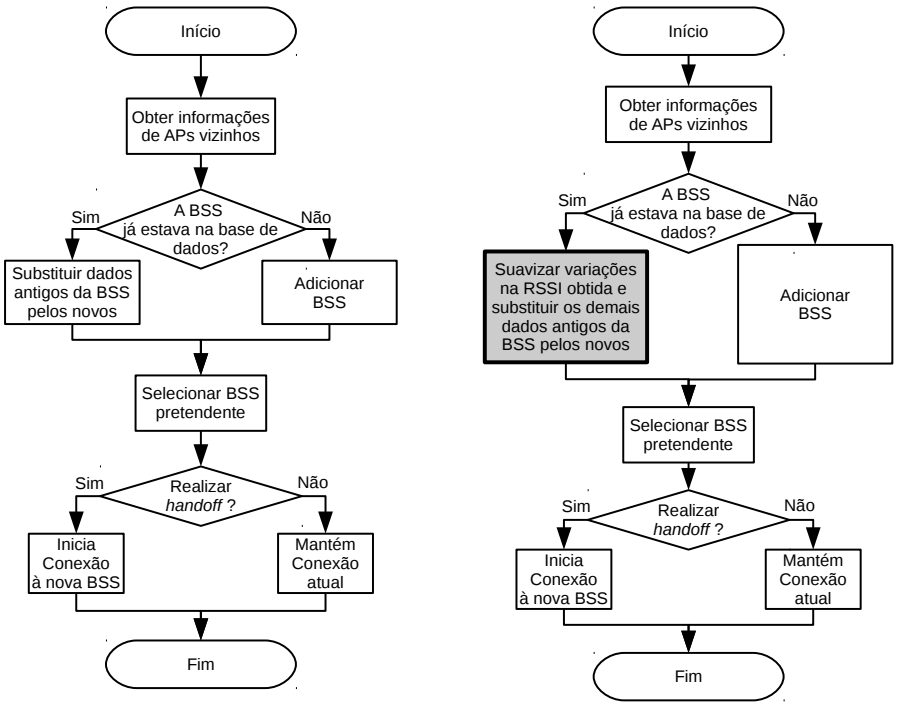


Figura 4. Fluxogramas de decisão de *handoff* (a) originalmente encontrado no *Wpa Supplicant*; (b) incluindo alterações para amenizar o problema do ping-pong.

Após obter informações sobre redes vizinhas, o *software* as armazena em uma base local da seguinte forma: caso um novo AP seja encontrado, ele é adicionado à base local; caso o AP já se encontre na base, os dados antigos são substituídos pelos dados atuais. A seguir, o *Wpa Supplicant* escolhe um AP pretendente dentre os pertencentes ao ESS ao qual a STA está atualmente associada. A escolha é realizada a partir de uma lista de prioridades, que é definida por parâmetros como segurança suportada e RSSI do AP.

Após escolher o AP pretendente, o *software* define se o *handoff* deve ou não ser realizado. Na versão atual do *software*, que é a 2.5, esta decisão é realizada com base no RSSI dos APs, incluindo o AP ao qual a STA está conectada, amostrado no decorrer da realização da varredura. Nesta implementação, o *handoff* é realizado somente caso o AP pretendente possua RSSI maior do que o do AP atual, e, além disso, a diferença entre os RSSIs deverá ser maior do que um determinado limiar. Este limiar varia de acordo com o RSSI recebido do AP atual no momento. Na implementação original do *Wpa Supplicant*, o maior limiar considerado é de 5 dB para um RSSI atual acima de -70 dBm, e o limiar diminui conforme o RSSI atual diminui.

Nesta implementação, os seguintes fatores favorecem a ocorrência de ping-pongs: (1) a nova informação de RSSI obtida a partir da varredura é copiada para a base de dados do *software* sem considerar o histórico do RSSI dos APs. Tendo em vista que o RSSI pode variar muito, conforme será mostrado adiante, *handoffs* indesejados podem ocorrer; e (2) o limiar que define a diferença entre os níveis dos sinais do AP atual e do AP pretendente não é suficiente para enquadrar a grande variabilidade do RSSI que pode ocorrer.

Para comprovar a grande variabilidade do RSSI, a Figura 5 mostra estatísticas

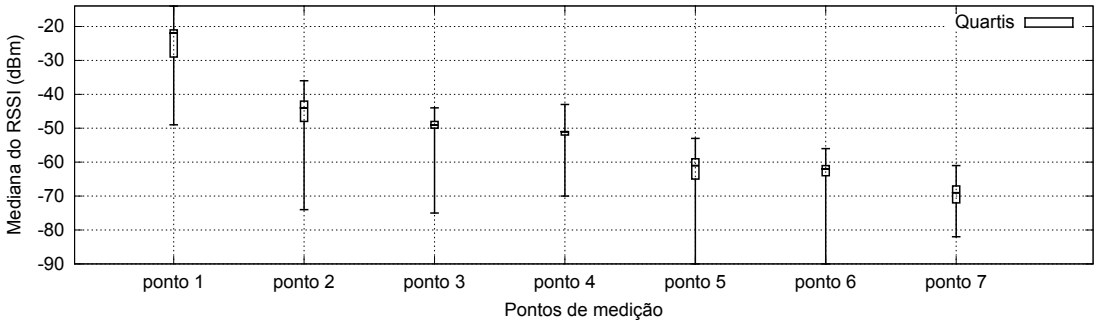


Figura 5. Mediana do RSSI (dBm) com estatísticas dos valores mínimo e máximo, e primeiro e terceiro quartis.

obtidas a partir de amostras de RSSI de *beacons* (em dBm, conforme reportado pelo cabeçalho *radiotap*), coletadas a diferentes distâncias de um ponto de acesso em um ambiente *indoor*. As amostras foram coletadas de forma consecutiva por uma mesma estação cliente em sete pontos de medição, sendo o ponto 1 o mais próximo do AP e o ponto 7 o mais distante. A estação cliente foi posicionada estaticamente em cada ponto para realizar a coleta de dez mil quadros de *beacon* consecutivos. O intervalo entre *beacons* foi de $0,1024$ segundos. Desta forma, a captura em cada ponto durou em torno de 17 minutos. A estação operou com uma interface sem fio *Broadcom BCM4313*, *driver brcmsmac*, e sistema operacional Ubuntu 14.04. O AP que emitiu os *beacons* possui modelo TL-WR740N e operou com Linux embarcado OpenWrt 14.07. O gráfico é um *Box Plot*, *i.e.*, mostra a mediana dos valores de RSSI para cada ponto de medição, primeiro e terceiro quartis (representados pelos retângulos inferior e superior à mediana, respectivamente), e valores máximo e mínimo (representados pelas linhas acima e abaixo da mediana, respectivamente). Através destes resultados é possível verificar a grande variabilidade do RSSI, chegando a 37 dB de diferença entre valores máximo e mínimo para o ponto 5. Esta variabilidade, que já foi observada em outros trabalhos, como o de [Mhatre e Papagiannaki 2006], pode ser explicada por alterações na propagação do sinal geralmente ocasionadas pela movimentação de pessoas e objetos em ambientes *indoor*.

Por não considerar a grande variabilidade do RSSI, o algoritmo de *handoff* utilizado pelo *Wpa Supplicant*, que é o *software* utilizado pela maioria das STAs na rede analisada neste trabalho, favorece a ocorrência de ping-pongs. Uma solução imediata, com o objetivo de mitigar a causa identificada para o problema, é a aplicação de uma técnica para suavizar as variações no RSSI amostrado. Um exemplo de técnica bem conhecida e amplamente utilizada para tal finalidade é a aplicação da Média Móvel Exponencialmente Ponderada (MMEP) ao conjunto de dados. A MMEP age sobre a série de dados como um filtro passa baixa, removendo ruídos de alta frequência e suavizando variações bruscas.

$$s_t = \alpha.x_t + (1 - \alpha).s_{t-1} \tag{1}$$

A Equação 1 descreve matematicamente a MMEP. Aplicando-se a MMEP ao problema do ping-pong, o conjunto de dados x representará os valores de RSSI amostrados a partir da varredura e o conjunto de dados s representará a MMEP destes valores, que serão efetivamente armazenados na base de dados e utilizados para a decisão de *handoff*. O momento $t = 0$ representa a inserção do AP na base de dados com seu valor original de RSSI amostrado. Conforme indica a equação, o parâmetro α define o peso que será dado

às amostras originais de RSSI. Desta forma, aumentando-se o valor de α, mais peso será dado às novas amostras, o que reduz a suavização dos dados. Reduzindo-se o valor de α, mais peso será dado às amostras antigas, aumentando-se a estabilidade dos dados. Para $\alpha = 1$, tem-se o algoritmo original utilizado pelo *Wpa Supplicant*.

A Figura 4(b) apresenta o fluxograma do algoritmo de *handoff* proposto neste trabalho, incluindo a aplicação da MMEP aos dados de RSSI. Em relação ao algoritmo tipicamente implementado nas STAs, apresentado na Figura 4(a), este novo algoritmo introduz a suavização das variações da RSSI antes de armazenar informações de varredura na base de dados.

4. Resultados dos testes comparativos

Para verificar o impacto da solução proposta no número de ping-pongs observados, testes comparativos foram realizados em uma rede real composta por dois pontos de acesso e uma STA posicionada em um local intermediário. Tanto os APs quanto a STA estavam em posição fixa. Inicialmente, a STA foi associada a um dos APs e informações foram coletadas no decorrer de uma semana, a partir de *logs* locais. Estas informações incluíram o momento em que a STA realizou varreduras e *handoffs*. A estação operou com interface sem fio *Broadcom BCM4313*, *driver brcmsmac*, e sistema operacional Ubuntu 14.04. A versão do *Wpa Supplicant* utilizada como base para os testes foi a 2.1. Os APs utilizados nos testes possuiam modelos TL-WR740N e TL-WR743ND e operaram com Linux embarcado OpenWrt 14.07.

Os testes foram realizados sequencialmente (em semanas distindas) com diferentes implementações do algoritmo de *handoff* sumarizadas na Tabela 2. Além do algoritmo original utilizado pelo *Wpa Supplicant*, foram testadas a versão proposta com valores de $\alpha = \{0{,}2; 0{,}4; 0{,}6; 0{,}8\}$ e uma versão semelhante à original, porém com limiar de diferença de sinal fixo em 10 dB, *i.e*, para que o *handoff* seja realizado, a diferença entre o RSSI do AP pretendente em relação ao AP atual deverá estar acima de 10 dB. O novo valor de 10 dB representa o dobro do valor máximo utilizado no algoritmo original do *Wpa Supplicant*.

Tabela 2. Algoritmos de *handoff* testados.

Original	Versão original do *Wpa Supplicant*.
Limiar fixo	*Wpa Supplicant* modificado para operar com limiar fixo de 10 dB.
MMEP	*Wpa Supplicant* modificado para aplicar MMEP aos valores de RSSI obtidos a partir da varredura, com peso $\alpha = \{0{,}2; 0{,}4; 0{,}6; 0{,}8\}$ para amostras novas.

A Figura 6 mostra o número de ping-pongs obtidos para cada algoritmo testado no decorrer de uma semana. Para definir a ocorrência de ping-pongs, o valor de $Xmax$ foi definido como 240 segundos, tendo como base a frequência da execução de varreduras pela STA. Tendo em vista que o intervalo entre varreduras foi, em mais de 89% das vezes, de 120 segundos, o valor de $Xmax$ foi dado como o dobro deste intervalo. O valor de $Nmin$ foi mantido em 2 e a definição do limiar $Zmax$ não foi necessária, já que o *log* fornecido localmente pelo *software* informa a ocorrência de fato dos *handoffs*. Os resultados mostram que o algoritmo original sofreu 20 ping-pongs, enquanto o algoritmo limiar fixo, sofreu 6 ping-pongs, ou seja, sofreu redução de 70%, mas não eliminou os ping-pongs. O algoritmo proposto, que implementa a MMEP para amenizar variações

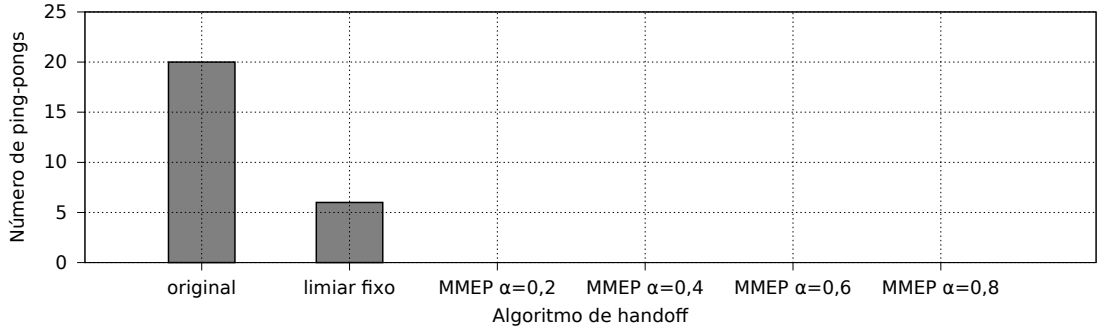


Figura 6. Número de ping-pongs observados, no período de uma semana, para cada algoritmo de *handoff* testado.

no RSSI dos APs vizinhos, não experimentou ping-pongs para os diferentes valores de α testados, ou seja, o ping-pong foi reduzido em 100% em relação ao algoritmo original.

Apesar da redução do número de ping-pongs observada para o algoritmo proposto, *handoffs* ainda foram observados para determinados valores de α. A Tabela 3 mostra, para cada algoritmo, a distribuição do número de *handoffs* por intervalo de tempo entre ocorrências. A última coluna apresenta o número total de *handoffs* observados, mostrando que este número reduziu com a redução do valor de α. Este resultado é esperado, tendo em vista que, quanto menor o valor de α, menor é o peso dado a novas amostras de RSSI, tornando mais eficaz a filtragem das variações bruscas e momentâneas da RSSI.

Tabela 3. Distribuição do número de *handoffs* de acordo com o intervalo de tempo entre ocorrências.

Algoritmo	0 e 5 min.	5 e 10 min.	10 e 15 min.	15 e 20 min.	20 e 25 min.	25 e 30 min.	acima de 30 min.	total
Original	62	29	15	12	5	10	37	170
Limiar fixo	21	10	6	3	1	2	21	64
MMEP $\alpha = 0{,}8$	9	7	2	1	4	4	27	54
MMEP $\alpha = 0{,}6$	0	3	0	0	0	0	6	9
MMEP $\alpha = 0{,}4$	0	0	0	1	1	0	3	5
MMEP $\alpha = 0{,}2$	0	0	0	0	0	0	0	0

As demais colunas da Tabela 3 apresentam o número de *handoffs* que ocorreram dentro de um determinado intervalo de tempo após a ocorrência anterior. Os intervalos estão distanciados em 5 minutos. Os resultados mostram que os *handoffs* observados para os algoritmos original e limiar fixo ocorrem com intervalos mais variados e com mais frequência em relação ao algoritmo MMEP. Observando-se pontualmente o *log* para os testes com o algoritmo proposto, pôde-se notar que a ocorrência de *handoffs* com intervalos maiores são comumente seguidos por outro *handoff* com um intervalo menor. Isto pode ser explicado pela ocorrência de variações bruscas na RSSI, tanto do AP atual ao qual a STA está associada quanto do AP pretendente. Nestes casos, a MMEP com valores de α mais altos não foi capaz de suavizar suficientemente estas grandes variações. No caso dos testes com $\alpha = 0{,}4$, por exemplo, a maior variação entre a RSSI amostrada (*i.e*, sem aplicação da MMEP) e a armazenada previamente na base de dados, para o mesmo AP, chegou a 11 dB. Já para $\alpha = 0{,}2$, nenhuma migração entre APs ocorreu.

5. Discussão

O processo de *handoff* pode ser dividido em quatro etapas. A primeira delas define quando a STA deve realizar a migração (etapa de *triggering*). A seguir, a STA realiza a descoberta de APs vizinhos e a escolha de um novo AP. Por fim, a STA realiza a migração [Mhatre e Papagiannaki 2006]. Alguns trabalhos, como o de [Mishra et al. 2003], caracterizam o *handoff* em termos de tempo necessário para sua realização e comportamento para diferentes fabricantes. Outros, como os de [Ramani e Savage 2005] e [Huang et al. 2006], buscam amenizar os inconvenientes ocasionados pelo *handoff*, através do emprego de técnicas para reduzir o tempo necessário para a realização de suas diferentes fases, como, por exemplo, a fase de descoberta e a fase da autenticação do cliente. O padrão IEEE 802.11r [IEEE 2008] também busca a realização de *handoffs* rápidos, e, para isso, utiliza métodos para reduzir o número de mensagens de autenticação trocadas durante a migração da STA. De fato, reduzir o tempo necessário para o *handoff* ameniza os efeitos do problema do ping-pong, já que o processo se tornaria menos custoso para a STA. Porém, não extingue o problema, *i.e.*, a conectividade seria momentaneamente interrompida e conexões ativas poderiam ser quebradas.

O trabalho [Raghavendra et al. 2007], que também aponta o problema da alta frequência de migrações em redes densas, realiza a análise dos *handoffs* em um cenário no qual muitos usuários se conectam ao mesmo AP simultaneamente. A principal causa apontada para o problema foi o alto índice de colisões, que levam à perda de quadros e à migração da STA para um novo AP em busca de melhor desempenho. Uma diferença em relação ao trabalho apresentado aqui é o fato de que a rede da UFF é menos densa, estando menos suscetível ao nível de colisões assumidos pelos autores. Além disso, como foi demonstrado na Seção 3, neste trabalho foi identificada a ocorrência de ping-pongs causados pela grande variação do RSSI, mesmo em cenários com baixa taxa de perdas. O trabalho de [Mhatre e Papagiannaki 2006] sugere o uso de um limiar de qualidade de conexão (denominado *operating region*) para evitar o ping-pong. A STA deve iniciar o processo de migração caso a qualidade esteja abaixo deste limiar. Porém, este limiar não soluciona o problema nos casos em que a STA está em um ambiente no qual a qualidade dos APs vizinhos é similar e abaixo do limiar. Além disso, o trabalho busca resolver um problema oposto ao observado aqui, que é a carência de realização de *handoffs* pelas STAs, mesmo em situações em que a qualidade da conexão poderia ser melhorada. Conforme foi mostrado, esta questão não é mais uma realidade. O trabalho de [Kim et al. 2012], assim como o apresentado aqui, também busca reduzir a frequência de *handoffs*, porém considera a trajetória de STAs em movimento. O problema descrito aqui ocorre também para estações com baixa mobilidade.

Por fim, outros trabalhos propõem melhorias no mecanismo de *handoff* e testam implementações utilizando o *Wpa Supplicant* e sistema Android. O trabalho de [Montavont et al. 2015], por exemplo, propõe um mecanismo para prever o nível de sinal futuro dos APs e antecipar a ocorrência da migração, resultando em melhor desempenho para estações em movimento.

5.1. Compromissos da solução proposta

A aplicação da MMEP para solucionar o problema do ping-pong possui a vantagem de amenizar variações bruscas no RSSI amostrado pela STA de APs nas redondezas, o que

reduz a ocorrência de *handoffs* indesejados. Entretanto, espera-se que sua aplicação possua a desvantagem de atrasar a reação da STA para casos em que o *handoff* é desejado, por exemplo, nos casos em que a estação está em movimento e uma migração seria benéfica, ou nos casos em que o RSSI do AP atual decaia em decorrência de um fator não momentâneo, como uma falha ou a instalação de um obstáculo. Nestes casos, é esperado que o *handoff* ocorra, porém com certo atraso em relação ao algoritmo original. Em trabalhos futuros, estudos serão realizados para verificar quantitativamente este atraso. Além disso, tendo em vista que a maioria dos dispositivos atualmente são *smartphones*, outra ideia interessante a ser avaliada é a utilização de dados obtidos a partir de sensores incluídos nos dispositivos, como o acelerômetro, por exemplo. Através destes dados, o mecanismo de decisão de *handoff* poderia ser otimizado nos casos em que a STA possui pouca mobilidade, por exemplo.

6. Conclusões

Este trabalho descreveu e discutiu o problema do ping-pong e analisou sua ocorrência em uma rede real densa em produção, composta por 390 pontos de acesso e aproximadamente 16.000 estações cliente. Os resultados mostraram que o problema é comum e recorrente para dispositivos de diferentes fabricantes, e, em particular, no sistema Android, que opera em mais de 60% destes dispositivos. Tendo em vista que este SO possui código acessível, este foi utilizado como base para a análise do problema e a proposta de uma solução. A análise da ocorrência do problema em outras redes será realizada em trabalhos futuros utilizando-se traços disponibilizados pela comunidade científica.

O *software* que comumente controla a realização de *handoffs* em dispositivos Android (e outras distribuições Linux) é o *Wpa Supplicant*. A análise de seu código mostrou que este *software* não considera a grande variação do RSSI amostrado de APs vizinhos em um ambiente *indoor* e, desta forma, permite que *handoffs* indesejados ocorram em momentos pontuais de variações bruscas do RSSI. A solução proposta para o problema sugere a suavização destas variações através da aplicação da Média Móvel Exponencialmente Ponderada (MMEP) ao RSSI. Os resultados dos testes realizados em uma rede real mostram que, ao implementar a solução proposta, o problema do ping-pong foi mitigado para diferentes valores de ponderação α utilizados. A MMEP possui o benefício de suavizar o RSSI amostrado, porém, pode ocasionar o atraso na realização do *handoff* em momentos nos quais ele é desejado, como em casos de falha do AP ao qual a STA está conectada ou no caso em que a STA está em movimento. Uma análise mais profunda das limitações da aplicação da MMEP, e de potenciais soluções, será realizada em trabalhos futuros. Outra questão que pode ser investigada é o motivo da variação da RSSI, que pode também ser ocasionada pela qualidade do *hardware*, por exemplo, e não apenas pela alteração nos padrões de propagação do sinal eletromagnético.

Por fim, como o código de outros sistemas operacionais não está disponível para avaliação, por serem proprietários, uma análise mais profunda não pôde ser realizada para estes sistemas. No entanto, tendo em vista que outros sistemas, como os da Apple, também apresentaram o problema do ping-pong em condições bastante similares às enfrentadas pelos dispositivos Android, é possível especular que a causa também seja relacionada à aplicação direta de amostras de RSSI na tomada de decisão de *handoff*. Neste caso, a solução proposta poderia também ser aplicada a estes dispositivos.

Referências

Huang, P.-J., Tseng, Y.-C. e Tsai, K.-C. (2006). A Fast Handoff Mechanism for IEEE 802.11 and IAPP Networks. Em *Vehicular Technology Conference*, volume 2, páginas 966–970.

IEEE (2008). IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition. *IEEE Std 802.11r-2008 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008)*.

IEEE (2012). IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*.

Kim, M., Liu, Z., Parthasarathy, S., Pendarakis, D. e Yang, H. (2012). Association control algorithms for handoff frequency minimization in mobile wireless networks. *Wirel. Netw.*, 18(5):535–550.

Linux Wireless Wiki (2015). Wpa Supplicant Linux documentation page. URL: `https://wireless.wiki.kernel.org/en/users/documentation/wpa_supplicant`.

Magalhães, L. C. S., Balbi, H., Corrêa, C., Valle, R. e Stanton, M. (2013). SCIFI–A Software-Based Controller for Efficient Wireless Networks. Em *6th UbuntuNet Alliance annual conference*.

Mhatre, V. e Papagiannaki, K. (2006). Using Smart Triggers for Improved User Performance in 802.11 Wireless Networks. Em *Proceedings of MobiSys*, páginas 246–259.

Mishra, A., Shin, M. e Arbaugh, W. (2003). An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. *SIGCOMM Comput. Commun. Rev.*, 33(2):93–102.

Montavont, N., Blanc, A., Navas, R., Kerdoncuff, T. e Castignani, G. (2015). Handover triggering in IEEE 802.11 networks. Em *WoWMoM*, páginas 1–9.

Raghavendra, R., Belding, E. M., Papagiannaki, K. e Almeroth, K. C. (2007). Understanding Handoffs in Large IEEE 802.11 Wireless Networks. Em *Proceedings of IMC*, páginas 333–338.

Ramani, I. e Savage, S. (2005). Syncscan: practical fast handoff for 802.11 infrastructure networks. Em *INFOCOM*, volume 1, páginas 675–684.

Stiller, N. (2015). MAC Vendor Lookup. URL: `http://www.macvendorlookup.com/`.

Tan, L. e Wang, N. (2010). Future internet: The Internet of Things. Em *ICACTE*, volume 5, páginas 376–380.

Vasudevan, S., Papagiannaki, K., Diot, C., Kurose, J. e Towsley, D. (2005). Facilitating Access Point Selection in IEEE 802.11 Wireless Networks. Em *Proceedings of IMC*, páginas 293–298.

Trilha Principal do SBRC 2016
Sessão Técnica 5
Sistemas Distribuídos

Um Serviço de Detecção de Falhas com QoS Auto-Ajustável para Múltiplas Aplicações Simultâneas

Rogério C. Turchetti[1,2], Elias P. Duarte Jr.[1], Luciana Arantes[3], Pierre Sens[3]

[1] Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR – Brasil

[2]CTISM - Universidade Federal de Santa Maria (UFSM)
Avenida Roraima, 1000 – 97105-900 – Santa Maria – RS – Brasil

[3]Laboratoire d'Informatique de Paris 6 / Université Pierre et Marie Curie, Paris, France

turchetti@redes.ufsm.br, elias@inf.ufpr.br,

{luciana.arantes, pierre.sens}@lip6.fr

Abstract. *Failure detectors are used to monitor the state of the processes of a distributed application. In order to provide information of whether a monitored process is correct or has failed, a failure detector makes assumptions about time delays in the system. In this work we propose a failure detector service called QoS-CFDS (Quality of Service-Configurable Failure Detection Service). QoS-CFDS service is self-tuning, i.e., the service adapts its monitoring parameters according the environment variations and applications needs. We propose two strategies to adjust the detector in accordance with the requirements provided by multiple concurrent applications. The service was implemented using SNMP and experimental results are presented showing the performance of the detector, in particular the advantagens of using the self-tuning strategies to address the requirements for concurrent applications.*

Resumo. *Detectores de falhas monitoram os estados de processos de uma aplicação distribuída, efetuando hipóteses temporais sobre atrasos no sistema e disponibilizando informações sobre os estados destes processos. Neste trabalho é proposto o serviço de detecção de falhas denominado QoS-CFDS (Quality of Service-Configurable Failure Detection Service). O serviço QoS-CFDS é auto-ajustável, isto é, adapta seus parâmetros de monitoramento conforme as variações percebidas no ambiente e de acordo com as necessidades de cada aplicação. São propostas estratégias para ajustar a QoS do detector de acordo com os requisitos fornecidos por múltiplas aplicações simultâneas. Um protótipo do serviço proposto foi implementado com o protocolo SNMP e resultados experimentais são descritos para o desempenho do detector, incluindo os benefícios dos ajustes realizados para atender requisitos das aplicações simultaneamente.*

1. Introdução

Detectores de falhas são blocos fundamentais para auxiliar no desenvolvimento de aplicações distribuídas tolerantes a falhas. Um detector provê informações a respeito dos estados dos processos de um sistema. Os detectores de falhas foram propostos por Chandra e Toueg [Chandra and Toueg 1996] e sua abstração possibilita encapsular as premissas

temporais dos sistemas assíncronos, e, dependendo das propriedades que oferecem, permitem resolver problemas que não poderiam ser resolvidos de forma determinística em tais ambientes [Fischer et al. 1985]. Um detector de falhas é não confiável pois pode cometer erros, suspeitando erroneamente de um processo que não está falho. Porém, ao se dar conta que se enganou, corrige o seu erro parando de suspeitar do respectivo processo. Do ponto de vista funcional, os detectores de falhas monitoram os processos efetuando hipóteses temporais sobre atrasos no sistema. Em geral, o monitoramento realizado pelos detectores de falhas é baseado em troca de mensagens. O processo monitorado envia periodicamente mensagens de *heartbeat*. Caso o detector de falhas não receba um *heartbeat* dentro de um limite de tempo especificado, o respectivo processo monitorado passa a ser considerado suspeito de ter falhado.

Ao longo dos anos, foram propostas aplicações de detectores de falhas em diferentes contextos: sistemas de larga escala [Hayashibara et al. 2002], redes sem fio [Greve et al. 2011], computação em nuvem [Xiong et al. 2012], redes definidas por software [Turchetti and Duarte Jr. 2015]). Diversos aprimoramentos do serviço de detecção também foram propostos, como a melhoria de desempenho [Fetzer et al. 2001], interfaces mais adequadas para as aplicações [Felber et al. 1999] e parâmetros que podem ser adaptados de acordo com mudanças no ambiente [Bertier et al. 2003, Dixit and Casimiro 2010, Xiong et al. 2012, Xiaohui and Yan 2014, Tomsic et al. 2015]. Neste sentido, uma característica importante é a capacidade do detector de prever o instante de tempo em que a próxima mensagem de um monitoramento será recebida. Em geral, as abordagens são baseadas em estimativas que visam corrigir o tempo de espera (*timeout*) de acordo com o comportamento do ambiente de execução.

Um desafio prático dos detectores de falhas é oferecer um serviço que possa ser também adaptado para monitorar processos de acordo com as necessidades de cada aplicação distribuída. Em particular, é importante considerar que existem aplicações que possuem restrições temporais distintas umas das outras [Chen et al. 2000]. As restrições temporais dizem respeito ao tempo necessário para a detecção de uma falha, ou ao tempo para a correção de uma falsa suspeita ou ainda, ao intervalo de tempo entre duas falsas suspeitas. Por exemplo, definir um tempo curto para a detecção de falhas pode ser uma boa estratégia para certas aplicações, mesmo que este serviço esteja sujeito a detectar falhas de forma incorreta. Por outro lado, para outras aplicações que acomodam maiores atrasos, ter um serviço que comete menor número de erros pode ser melhor, mesmo que o tempo para a detecção de falha tenha um período mais longo. Portanto, é relevante oferecer um serviço capaz de se adaptar às necessidades de cada aplicação; há poucas soluções que abordam este problema [de Sá and de Araújo Macêdo 2010].

Chen et al. [Chen et al. 2000] propuseram um grupo de métricas que permitem avaliar a qualidade de serviço (QoS - *Quality of Service - QoS*) oferecida pelos algoritmos de detecção de falhas. Bertier et al. [Bertier et al. 2003] utilizam o algoritmo proposto por Chen para fornecer um serviço para detecção de falhas que pode ser compartilhado por diversas aplicações com base em suas restrições temporais. O presente trabalho tem por objetivo propor um serviço para detecção de falhas com QoS "auto-ajustável". Isto é, o serviço adapta seus parâmetros de monitoramento conforme as variações percebidas no ambiente (como atrasos de comunicação e sobrecarga de CPU) e de acordo com as necessidades de cada aplicação. O serviço proposto é denominado de QoS-CFDS (*QoS-*

Configurable Failure Detection Service) e pode ser compartilhado por múltiplas aplicações simultaneamente, fornecendo as garantias de qualidade de serviço requisitadas.

Para garantir parâmetros de QoS definidos pelas aplicações, o QoS-CFDS implementa uma estratégia que permite que aplicações especifiquem seus requisitos, por exemplo, uma aplicação pode requisitar um limite de tempo para a detecção de uma falha e o QoS-CFDS ajusta os parâmetros de monitoramento do detector de falhas com base nestas requisições de entrada. Em outras palavras, com base nos parâmetros fornecidos e no comportamento observado da rede, o QoS-CFDS é configurado para tentar satisfazer as solicitações das aplicações. Um algoritmo que recebe os valores de QoS fornecidos por uma aplicação e ajusta os parâmetros de monitoramento do detector, é proposto.

Uma contribuição deste trabalho é justamente propor, implementar e avaliar duas estratégias que permitem compartilhar o serviço de detecção de falhas com QoS entre diversas aplicações simultâneas. Em outras palavras, as estratégias ajustam a periodicidade de envio (representado por η) das mensagens de *heatbeat* que são carregadas via protocolo SNMP, para tentar cumprir com todas as requisições de QoS. A estratégia denominada de η_{max} busca encontrar um valor amplo o suficiente para acomodar todos os requisitos especificados pelas múltiplas aplicações, ao passo que, a estratégia denominada de η_{GCD} busca encontrar um máximo divisor comum dos requisitos de entrada para ser atribuído a η. Os resultados dos cômputos das estratégias definem um valor para η que pode ser aplicado em diferentes propósitos (e.g. considerando o tempo de detecção) e para diferentes ambientes de execução (e.g. considerando os atrasos na comunicação).

Um protótipo do serviço proposto foi implementado utilizando o protocolo SNMP (*Simple Network Management Protocol*) [Harrington et al. 2002]. Uma MIB (*Management Information Base*) denominada $fdMIB$ (*failure detection*) que permite a integração do serviço com as aplicações distribuídas e os processos a serem monitorados, é proposta. A $fdMIB$ armazena informações importantes sobre todos os componentes da arquitetura, como também executa a própria atividade de monitoramento dos processos. Resultados experimentais obtidos mostram o desempenho do QoS-CFDS, em especial os benefícios de oferecer o serviço para múltiplas aplicações simultâneas.

O restante do trabalho está organizado da seguinte forma. A Seção 2 traz uma breve introdução aos detectores de falhas e apresenta o modelo de sistema considerado neste trabalho. Na Seção 3 é apresentado o processo para a configuração de *timeout* adaptativo, como também estratégias para configurar o detector de falhas de acordo com as necessidades de múltiplas aplicações simultâneas. Na Seção 4 é descrita a arquitetura, implementação e resultados experimentais do QoS-CFDS. Por fim, a conclusão segue na Seção 5.

2. Detectores de Falhas e Modelo de Sistema

Os detectores de falhas foram propostos por Chandra e Toueg [Chandra and Toueg 1996] como uma abstração que possibilita encapsular o indeterminismo temporal dos sistemas assíncronos. O indeterminismo é a raiz da impossibilidade FLP [Fischer et al. 1985] e suscita que, devido à falta de conhecimento temporal dos sistemas assíncronos, no caso da falha de algum processo, não há algoritmos determinísticos que possam garantir a resolução do consenso. Por trás ao problema está a impossibilidade em determinar quando um processo está falho ou muito lento, devido a fatores como, por exemplo, aumento no

tráfego da rede ou sobrecargas de CPU.

Sendo assim, um detector de falhas permite encapsular o indeterminismo por indicar os estados de cada processo participante do sistema. Os estados indicam se um processo está ou não suspeito, de acordo com o monitoramento que ocorre por troca de mensagens e respeitando um limite de tempo (*timeout*). Um detector de falhas pode ser acessado por um processo p_i para obter informações referentes aos estados de outros processos. A informação retornada por um detector de falhas pode ser incorreta, isto é, um processo não falho pode ser considerado suspeito e vice-versa. Além disso, um detector de falhas pode fornecer informações inconsistentes divergindo de outros detectores.

Chandra e Toueg classificam os detectores de falhas através de duas propriedades: (1) completude (*completeness*) e (2) precisão (*accuracy*). Informalmente, a completude caracteriza a capacidade do detector de falhas suspeitar de todos os processos falhos (*crashed*), enquanto que a precisão caracteriza a capacidade do detector de falhas não suspeitar de processos não falhos (corretos). Estas propriedades permitem investigar os problemas que podem ser resolvidos com o uso de detectores, ou quais propriedades devem ser garantidas por um detector para resolver um determinado problema.

Modelo de Sistema

Neste trabalho considera-se um sistema distribuído assíncrono composto por um conjunto Π de n processos/hosts, incrementado com detector de falhas [Chandra and Toueg 1996]. Os processos falham por parada total (*crash*), ou seja, deixam de executar suas tarefas prematuramente. O monitoramento dos processos, em cada rede local, ocorre por um detector de falhas não confiável, o qual nunca falha. O monitoramento dos processos é realizado através de mensagens de *heartbeat* enviadas periodicamente pelos processos monitorados ao detector. O monitoramento resulta nos seguintes estados: um processo é considerado *suspeito* quando uma mensagem de *heartbeat* não é recebida dentro do intervalo de tempo esperado; *não suspeito* ou *correto* ocorre quando a mensagem de *heatbeat* é recebida pelo detector de falhas dentro do intervalo de tempo estabelecido. A comunicação é realizada por troca de mensagens através de um canal confiável, isto é, o canal não cria, não duplica, não perde e não altera mensagens de controle.

3. QoS-CFDS: Adaptação do Timeout e Configuração dos Parâmetros de QoS

Indiscutivelmente, uma das funções mais importantes executadas pelos detectores de falhas é o cálculo do intervalo de *timeout*. Uma abordagem comumente utilizada é a adaptativa [Bertier et al. 2003, Dixit and Casimiro 2010, Xiong et al. 2012, Xiaohui and Yan 2014]. O serviço para detecção de falhas proposto neste trabalho calcula o intervalo de *timeout* com base na estimativa do tempo de chegada da próxima mensagem de *heartbeat*, somado a uma margem de segurança adaptativa. Considere dois processos: p_i e p_j, onde p_j monitora p_i. A cada intervalo de tempo η (η representa a periodicidade de envio de mensagens), p_i envia uma mensagem de *heartbeat* para o monitor p_j. Sejam m_1, m_2,...,m_k as mensagens de *heartbeat*, m_k é a mais recente, recebidas por p_j. Sejam A_1, A_2,...,A_k os instantes de tempo nos quais as mensagens de *heartbeats* foram recebidas por p_j de acordo com o seu relógio local. EA pode ser estimado da seguinte forma:

$$EA_{k+1} = \frac{1}{k}\left(\sum_{i=1}^{k} A_{(i)} - \eta_i\right) + (k+1)\eta \quad (1)$$

EA é o instante de tempo estimado para a chegada da próxima mensagem de *heartbeat*. Sendo assim, o valor para o instante de tempo em que o próximo *timeout* (τ) irá expirar, é calculado da seguinte forma:

$$\tau_{(k+1)} = EA_{(k+1)} + \alpha_{(k+1)} \quad (2)$$

Para tornar a detecção de falhas mais precisa a margem de segurança α é calculada de acordo com o algoritmo de Jacobson [Jacobson 1988], que trabalha com base nos instantes de tempo em que as mensagens de *heartbeat* (hb) são recebidas, isto é, $diff_{(k)}$ apresentado na expressão 3, representa a diferença entre a '*k-ésima*' e a '*k-ésima*-1' mensagens:

$$diff_{(k)} = hb_k - hb_{k-1} \quad (3)$$

Na expressão (4), $delay_{(k+1)}$ é o valor do atraso que é calculado em termos da diferença $diff_{(k)}$, e γ representa o peso das novas amostras, Jacobson sugere o valor de $\gamma = 0.1$:

$$delay_{(k+1)} = (1-\gamma) \cdot delay_{(k)} + \gamma \cdot diff_{(k)} \quad (4)$$

$var_{(k+1)}$ representa a variação da diferença:

$$var_{(k+1)} = (1-\gamma) \cdot var_{(k)} + \gamma \cdot (|diff_{(k)} - var_{(k)}|) \quad (5)$$

Por fim, a margem de segurança $\alpha_{(k+1)}$ é obtida através da expressão 6, onde β e ϕ são constantes que tipicamente recebem os seguintes valores [Bertier et al. 2003]: $\beta = 1$, $\phi = 4$ e $\gamma = 0.1$.

$$\alpha_{(k+1)} = \beta \cdot delay_{(k+1)} + \phi \cdot var_{(k+1)} \quad (6)$$

Com base nos cálculos apresentados nesta seção, o intervalo de *timeout* τ é ajustado a cada nova mensagem de *heartbeat* recebida pelo detector de falhas.

Configurando o Detector de Falhas com Base nos Parâmetros de QoS

Um desafio dos detectores de falhas é oferecer um serviço que possa ser adaptado de acordo com as necessidades de cada aplicação. Nesta seção é proposto um algoritmo que permite configurar o detector de falhas de acordo com as necessidades de QoS de cada aplicação. Inicialmente, as funcionalidades do algoritmo são apresentadas considerando requisições de uma única aplicação. Subsequentemente, o algoritmo é estendido para funcionar com duas estratégias (denominadas η_{max} e η_{GCD}) que trabalham de acordo com as requisições de QoS para múltiplas aplicações. Desta forma, mesmo se múltiplas aplicações usarem o detector de falhas para monitorar o mesmo processo, a estratégia deve suportar diferentes requisitos de QoS, e um único intervalo de *heartbeat* deve ser calculado, satisfazendo todas as requisições.

Uma aplicação entra com seus requisitos de QoS através dos seguintes parâmetros, que representam as métricas primárias propostas por Chen et al. [Chen et al. 2000]:

- T_D^U: limite máximo para o tempo de detecção;
- T_M^U: limite máximo para a duração de um erro;
- T_{MR}^L: limite mínimo para a ocorrência entre dois erros consecutivos.

A partir destes parâmetros, o detector de falhas encontra um valor apropriado para η, como descrito a seguir inicialmente para uma aplicação e depois para múltiplas aplicações.

Ativando QoS: *Uma Única Aplicação*

Para que o serviço de detecção de falhas possa ativar a QoS, a aplicação requisitante deve entrar com os parâmetros descritos acima. Além disso, três outros parâmetros são necessários e providos pelo detector de falhas: probabilidade de perda de mensagens (p_L), variância do atraso ($V(D)$) e estimativa do tempo da próxima mensagem de *heartbeat* (EA). A probabilidade de perda de mensagens é encontrada calculando $p_L = L/k$, onde k é o número total de mensagens que foram enviadas e L é o número de mensagens perdidas. $V(D)$ é calculado observando a variância nos tempos das mensagens recebidas. Uma aplicação cliente envia os dados de entrada que são processados através de dois passos apresentados no Algoritmo 1. O Algoritmo 1, adaptado de [Chen et al. 2000], tem por objetivo encontrar um valor apropriado para η de forma a atender as necessidades de cada aplicação.

O Algoritmo 1 é composto por dois passos. No passo 1 as aplicações fornecem seus parâmetros de QoS e os valores de p_L e $V(D)$ são obtidos na linha 3 e, a seguir, o menor valor de η_{max} é obtido (linha 7 ou linha 9), dependendo de qual dos valores é menor: $min(\gamma * T_M^U, T_D^U)$. Vale ressaltar que existem duas condições em que a QoS não pode ser ativada, sendo elas: (i) se no passo 1 o valor de η_{max} for igual a zero ou (ii) se no passo 2 o intervalo de *heartbeat* η não for obtido. Entretanto, se no passo 1 $\eta_{max} \geq 0$, então a linha 10 do algoritmo é executada invocando o passo 2.

O objetivo do passo 2 é encontrar um valor para η que cumpra com os requisitos da aplicação. Como η deve ser maior do que zero e menor do que η_{max}, propomos inicializar η igual a η_{max} (linha 16). A cada novo valor definido para η, a seguinte condição é testada $f_\eta \geq T_{MR}^L$ (linha 17). Enquanto esta condição não for verdadeira, f_η é recalculado e um novo teste é executado.

Algorithm 1: Algoritmo para configuração de QoS.

```
1  Step1 (T_D^U, T_M^U, T_MR^L)                          ▷ no Step1 calcula-se o valor o valor para η_max
2
3    γ ← (1 − p_L)(T_D^U)^2/(V(D) + (T_D^U)^2);
4
5    if ((γ > 0) and (T_D^U > 0) and (T_M^U > 0)) then
6        if ((γ * T_M^U) > T_D^U) then
7            η_max ← T_D^U;
8        else
9            η_max ← γ * T_M^U;
10       run Step2(T_D^U, T_MR^L, η_max)
11   else
12       return ("QoS não pode ser ativada");

13 Step2 (T_D^U, T_MR^L, η_max)                          ▷ no Step2 calcula-se o valor para η
14
15   f_n ← 1;
16   η ← η_max;                                          ▷ η_max é utilizado como valor para inicializar η
17   while (fη < T_MR^L) do
18       for (j ← 1; j ≤ T_D^U/η; j++) do
19
20           f_n ← f_n * (V(D)+(T_D^U − jη)^2)/(V(D)+(p_L(T_D^U − jη)^2));
21
22       f_n ← f_n * η;
23       η ← η − (η * 0.01)                              ▷ para o valor de fη crescer reduzimos o valor de η;
24   return (η);
```

Para cada valor de η, f_η é calculado em sucessivas iterações com j=1 até $j > T_D^U/n$ (linhas 18 e 20). Note que, quando η diminui, f_η aumenta, o oposto também é verdadeiro. Por esta razão, sempre reduzimos o valor de η (linha 23, reduz em 1% o valor de η) até que $f_\eta \geq T_{MR}^L$. Neste ponto o valor de η é finalmente encontrado. Vale destacar que este é o maior valor possível que permite cumprir com os parâmetros de QoS fornecidos pela aplicação, assim pode ser usado um η menor, mas que gera mais mensagens de monitoramento.

Ativando QoS: *Múltiplas Aplicações*

Para encontrar um valor para η que satisfaça aos requisitos de múltiplas aplicações, são propostas duas diferentes estratégias, η_{max} e η_{GCD}, descritas a seguir. A proposta η_{max} busca um valor do intervalo de monitoramento grande o suficiente para acomodar todos os requisitos. A proposta η_{GCD} parte do princípio que, se um processo envia uma mensagem de *heartbeat* a cada x unidades de tempo, ele também pode enviar um *heartbeat* a cada y unidades de tempo, se x divide y.

Estratégia η_{max}: ao invés de atribuir $\eta_{max} = min(\gamma T_M^U, T_D^U)$ para uma única aplicação, para n aplicações $\eta_{max} = min(\gamma T_{M1}^U, T_{D1}^U, \gamma T_{M2}^U, T_{D2}^U, ..., \gamma T_{Mn}^U, T_{Dn}^U)$, onde $i = 1...n$, T_{Mi}^U e T_{Di}^U correspondem aos requisitos da aplicação i (App_i). Em outras palavras, $\eta_{max} = min(\eta_1, \eta_2, ..., \eta_n)$, Com esta única modificação, é possível usar o Algoritmo 1 para calcular o valor apropriado de η.

Estratégia η_{GCD}: esta estratégia calcula o máximo divisor comum (*Greatest Common Divisor - GCD*) entre todos os η_i, onde $i = 1...n$, e n é o número aplicações.

Inicialmente, para cada η_i um novo η'_i é encontrado da seguinte forma: $\eta'_i = 2^n$ onde $2^n < \eta_i$ e $\eta'_i \in \mathbb{Z}^+$. Desta forma, $\eta_{GCD} = GCD(\eta'_i, ..., \eta'_n)$. Se $\eta_{GCD} > 0$, o intervalo de monitoramento fica definido; caso contrário esta estratégia não pode ser aplicada. Observe que o valor de η_{GCD} é sempre menor do que o valor da estratégia η_{max}. Neste sentido, esta estratégia resulta em um número maior de mensagens na rede. Por outro lado, há benefícios como a redução do tempo de detecção de falhas (T_D).

Vale ressaltar que, utilizando múltiplas aplicações, a proposta apresentada se adapta mesmo em condições em que ocorre violação de QoS causadas por atrasos na comunicação. Por exemplo, se o QoS-CFDS detectar que não é possível sustentar os valores mínimos de QoS requeridos, ele reajusta o valor de η aumentando para um outro valor pré-estabelecido, caso haja. Por exemplo, considere duas aplicações utilizando a estratégia η_{max}, o valor final é $\eta_{max} = min(\eta_1, \eta_2)$, se η_1 foi escolhido por ser o menor e caso este valor não cumpra com os requisitos de QoS, o próximo valor utilizado será η_2. Nesse caso o serviço garante, mesmo que de forma parcial, a QoS requerida para determinadas aplicações.

4. Implementação e Resultados Experimentais

O serviço QoS-CFDS foi implementado usando o protocolo de gerência de redes da Internet, Simple Network Management Protocol (SNMP) [Harrington et al. 2002]. A Figura 1 apresenta a arquitetura do QoS-CFDS. As aplicações (App) podem se registrar no *Host Monitor* para receber informações sobre os estados dos processos que executam em um *Host Monitorado*. O *Host Monitor* configura um *timeout* adaptativo (τ), como descrito na seção anterior, em cada *Host Monitorado*, que encaminha *heartbeats* ao *Host Monitor* periodicamente com intervalo η. São utilizadas mensagens SNMP para transportar as mensagens de *heartbeat*.

Conforme pode ser visto na Figura 2, para uma aplicação encaminhar seus valores de QoS, algumas informações são repassadas por parâmetro, como exemplo: endereço IP do *Host Monitor*, o identificador do objeto onde os dados são armazenados (*Object Identifier* - OID), os requisitos de QoS, entre outras informações. Além das informações de QoS, o componente *Estimator* é responsável por realizar cálculos estatísticos (p_L, $V(D)$ e EA) e o componente *Configurator* executa o Algoritmo 1.

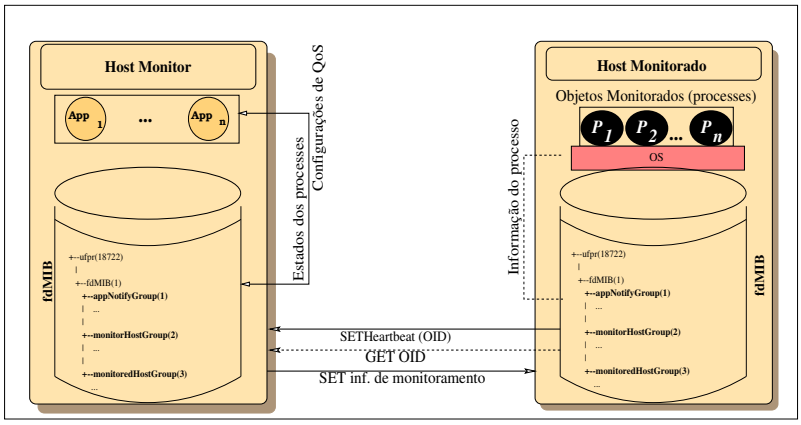


Figura 1. Arquitetura do QoS-CFDS.

Uma MIB ($fdMIB$) implementada no *Host Monitor* executa o monitoramento

dos estados dos processos e armazena informações a respeito de todos os componentes existentes na arquitetura. A $fdMIB$ consiste de três grupos, `appNotifyGroup`, `monitorHostGroup` e `monitoredHostGroup`, descritos a seguir.

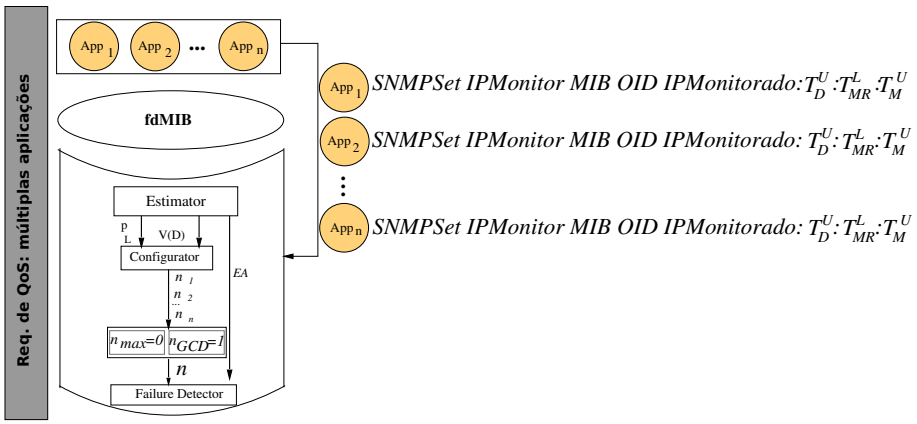


Figura 2. A aplicação configura os parâmetros de QoS na $fdMIB$.

O grupo `appNotifyGroup` é responsável por manter informações sobre os parâmetros de QoS fornecidos pelas aplicações. Cada aplicação que desejar receber informações sobre os estados dos processos deve informar, além do seu endereço de IP e porta, os seguintes parâmetros: T_D^U, T_M^U e T_{MR}^L. Se existirem múltiplas aplicações, é possível definir qual estratégia utilizar η_{max} η_{GCD}. Se o objeto não é setado, η_{max} é utilizada por default. O `appNotifyGroup` também possui um objeto denominado de `receiveHB`, usado para controlar as mensagens de *heartbeat* recebidas dos processos monitorados. A cada mensagem de *heartbeat* recebida, a $fdMIB$ atualiza o estado do correspondente processo. A mensagem é identificada pelo seu OID (*Object IDentifier*), atualizando o valor do objeto que pertence ao `monitorHostGroup`. Por fim, para aplicações registradas na $fdMIB$, é possível enviar notificações de mudanças de estados dos processos via SNMP *traps* (`notifyTrap`).

O grupo `monitorHostGroup` mantém informações sobre os processos monitorados, incluindo: endereço IP, porta e estado do processo (0=não suspeito, 1=suspeito). Outras informações incluem: número de falsas suspeitas, tempo de detecção da falha, intervalo de *heartbeat*, cálculos estatísticos conforme apresentado na Figura 2, entre outras. No `monitorHostGroup` são gerenciadas todas as atividades de monitoramento, em particular o gerenciamento e controle das *threads* responsáveis por implementarem o *timeout* para cada processo monitorado.

O grupo `monitoredHostGroup` é responsável por enviar periodicamente as mensagens de *heartbeat* ao *Host Monitor* sendo, portanto, executado pelo *Host Monitorado*. Com a finalidade de se comunicar com o *Host Monitor*, as seguintes informações são mantidas pelos objetos: endereço IP do monitor, OID do próprio *Host Monitorado* (o OID é usado para identificar o *Host Monitorado* na MIB) e o intervalo de *heartbeat*. Vale lembrar que η é calculado no *Host Monitor* de acordo com as requisições de QoS, para então ser enviado ao *Host Monitorado*, de forma a configurá-lo para encaminhar mensagens na periodicidade indicada.

4.1. Avaliação Experimental do QoS-CFDS

Nesta seção são apresentados experimentos executados com o objetivo de avaliar o serviço para detecção de falhas proposto. Os experimentos foram executados em uma rede local. Com a finalidade de avaliar o QoS-CFDS trabalhando com diferentes requisições de QoS, três aplicações com seus respectivos parâmetros de QoS são consideradas e apresentadas na Tabela 1. Por exemplo, considerando a App_1 os seguintes valores de QoS são requisitados: uma falha deve ser detectada no período máximo de 8 segundos (T_D^U=8000ms), o detector de falhas corrige um erro no limite máximo de 1 minuto (T_M^U=60000ms) e o detector de falhas comete no máximo 1 erro por mês (T_{MR}^L=2592000000ms).

Tabela 1. Parâmetros de QoS de cada aplicação.

Aplicações	T_D^U (ms)	T_M^U (ms)	T_{MR}^L (ms)
App_1	8000	60000	2592000000
App_2	14000	120000	2592000000
App_3	16000	240000	2592000000

Com base no Algoritmo 1, nos dados de entrada apresentados na Tabela 1 e considerando as estratégias para o cômputo do η, tem-se os seguintes resultados: η_{max}= $min(1.954467, 3.901890, 4.694764) \rightarrow \eta$=1.95 e η_{GCD}= $GCD(1, 2, 4) \rightarrow \eta$=1.

Nos experimentos p_L=0.01 e $V(D)$=0.02. Além disso, como apresentado na Expressão 1, o cálculo de EA utiliza um histórico das últimas mensagens de *heartbeat* recebidas, o tamanho da janela é indicado nos experimentos.

Resultados Experimentais

Para os experimentos foram utilizadas as seguintes máquinas físicas conectadas em rede Ethernet 100Mbps: processador Intel Core i5 CPU 2.50GHz com 4 núcleos e sistema operacional Ubuntu 12.04.4 executando a $fdMIB$ como monitor, o *host* monitorado possui um processador Intel Core i5 CPU 3.20GHz com 4 núcleos executando o sistema operacional Ubuntu 13.10, com kernel 3.2.0-58. A MIB foi projetada com o *toolkit* Net-SNMP[1], versão 5.4.4.

Nos gráficos das Figuras 3, 4(a) e 4(b) são ilustrados os envios de 200 mensagens de *heartebeat* que ocorrem entre o monitor e o *host* monitorado. Durante as primeiras 50 mensagens, o intervalo de envio de mensagens de *heartbeat* é configurado igual a 1000ms. Este valor justifica-se por ser um valor que contempla tanto a estratégia η_{max}=1.95s, quanto a η_{GCD}=1s. Após o envio de 50 mensagens, o intervalo de *heartbeat* é alterado para 5000ms. Novamente, este valor justifica-se por não contemplar nenhuma das estratégias, cujos máximos são: $\eta_{max} \leq$ 4.69s e $\eta_{GCD} \leq$ 4s.

Além disso, essa mudança no valor de η é para verificar o comportamento do serviço de detecção de falhas através da simulação de sobrecarga nos processos ou canais de comunicação (por exemplo, quando o intervalo de *heartbeat* passa de 1000ms para 5000ms, esse atraso poderia ser causado por alguma sobrecarga), como também o comportamento do QoS-CFDS considerando os parâmetros de QoS.

[1]http://www.net-snmp.org

Para calcular o EA, o tamanho da janela deslizante é definido para 5 mensagens. Pode-se observar na Figura 3 que há um tempo inicial para a estabilização do *timeout*. Este mesmo cenário pode ser observado quando ocorre a mudança no intervalo de emissão de η=1000ms para η=5000ms. Além disso, devido a esta mudança brusca no valor de η, em que o *timeout* atinge o maior valor, é possível observar na Figura 4(a) que ocorrem 6 falsas detecções. Estas podem ser observadas através do parâmetro de T_M. A Figura 4(a), também apresenta as 3 aplicações com seus respectivos parâmetros de QoS (vide Tabela 1), onde cada aplicação especifica seu próprio T_D^U. Com base nestes parâmetros, é possível observar, na Figura 4(a), que o tempo T_D calculado nos experimentos é sempre menor do que T_D^U especificado pela App_1, T_D^U da App_2 e T_D^U especificado pela App_3. Por esta razão, as falsas suspeitas detectadas e mostradas na Figura 4(a), não são notificadas para as aplicações, mesmo quando η atinge o valor de 5.

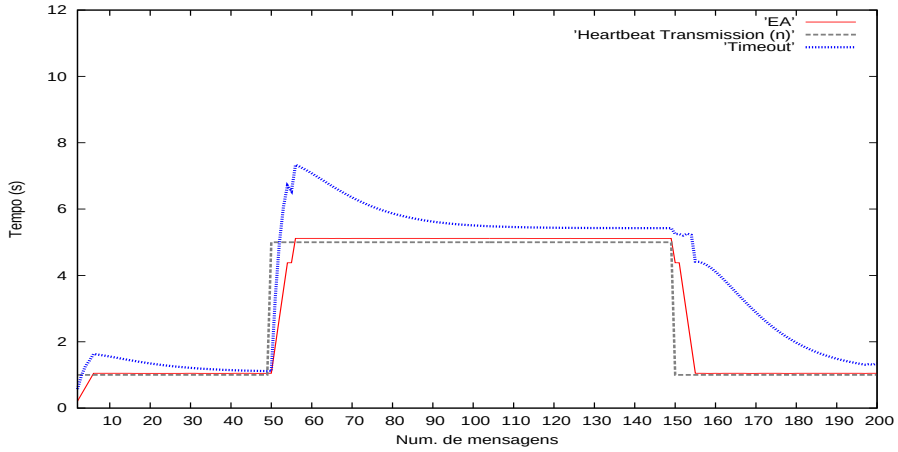
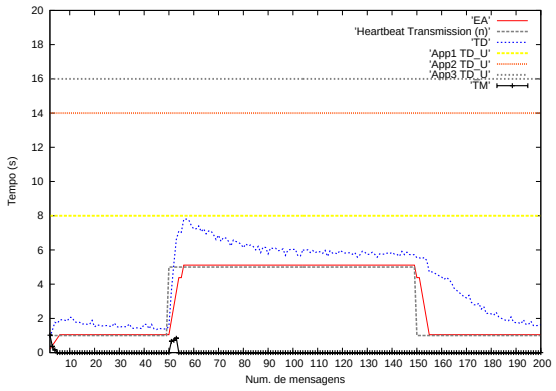


Figura 3. *Timeout* **adaptativo para diferentes intervalos de envio de** *heartbeats*.

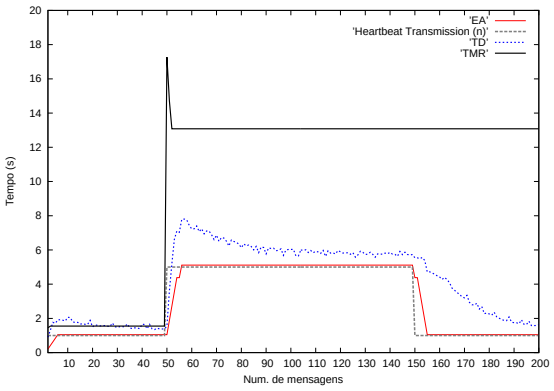
Uma situação similar ocorre na Figura 4(b), onde a média do T_{MR} é de 13086ms, considerando as falsas detecções, de acordo com as requisições de QoS, T_{MR} deveria ser maior ou igual ao valor especificado pelas aplicações, isto é, T_{MR}^L=2592000000ms. Entretanto, como foi observado na Figura 4(a), nenhuma das 6 falsas suspeitas foram reportadas para as aplicações ($T_D \leq \min({T_D}^U{}_1, {T_D}^U{}_2, {T_D}^U{}_3)$), por esta razão, o valor do T_{MR} é 0 e não existem parâmetros de QoS que foram violados.

Na Figura 5 é avaliada a utilização de CPU e memória do contexto do *Host Monitor*. Para sobrecarregar o uso destes recursos pelo serviço de monitoramento do detector de falhas, os experimentos foram executados com diferente número de objetos na MIB, sendo que cada objeto corresponde a um processo monitorado, e para cada processo monitorado são escalonadas, periodicamente, *threads* responsáveis pelo controle do *timeout*. Um único *host* físico é responsável por enviar mensagens de *heartbeat*, identificando cada um dos processos monitorados inclusos na MIB. Os experimentos envolvem a inclusão de cada um dos objetos na $fdMIB$ e o recebimento das mensagens de *heartbeat*, o intervalo de envio de *heartbeat* é de 1ms. Os experimentos são executados no intervalo de uma hora, variando o número de objetos para monitoramento. O uso da memória ilustrado na Figura 5(a), cresce quase linearmente até 400 objetos, havendo uma estabilização na utilização de memória, não chegando a atingir 0.2%.

Quanto à utilização de CPU, pode-se observar na Figura 5(b) que cresce quando o

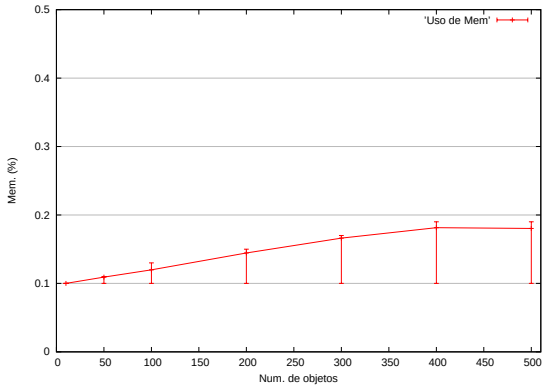


(a) T_D (Tempo de detecção) e T_M (Tempo de duração de um erro).

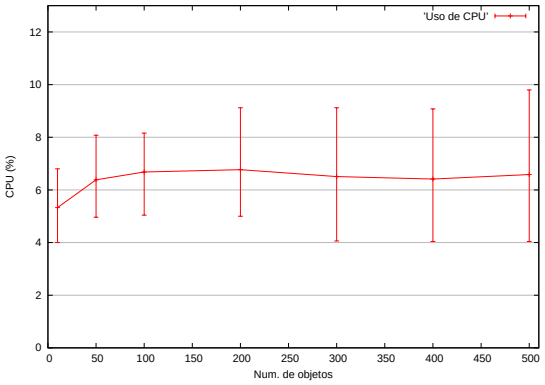


(b) T_{MR} (Tempo para recorrência ao erro).

Figura 4. Os parâmetros de QoS não são violados.



(a) Utilização de memória.



(b) Utilização de CPU.

Figura 5. Desempenho da $fdMIB$ no aumento de objetos de monitoramento.

número de objetos aumenta até 100, permanecendo quase constante para os experimentos em que o número de objetos cresce de 100 até 500, evidenciando que o serviço proposto é escalável. Por outro lado, pode-se observar que os valores máximos para a utilização de CPU apresentam um crescimento quase linear, estes picos são resultantes do registro inicial dos objetos na $fdMIB$. Entretanto, vale ressaltar que o procedimento para o registro de um processo na $fdMIB$, é necessário somente na primeira vez em que for monitorado.

Para comparar as estratégias η_{max} e η_{GCD} propostas neste trabalho, utiliza-se os parâmetros apresentados na Tabela 1, na qual os resultados obtidos para η foram: 1.95s e 1s, respectivamente. A Tabela 2 apresenta um comparativo entre as estratégias η_{max} e η_{GCD}. Para executar este experimento é utilizada a métrica P_A que corresponde à probabilidade de uma resposta exata, ou seja, é a probabilidade com que detectores de falhas geram saídas corretas em instantes de tempo aleatórios. P_A pode ser calculada da seguinte

forma:

$$P_A = 1 - \frac{E(T_M)}{E(T_{MR})} \tag{7}$$

O experimento mostrado na Tabela 2 foi executado durante o período de 1 hora, onde duas falhas foram simuladas através da omissão de duas mensagens de *heartbeat*, forçando o estouro do *timeout*. É possível verificar que η_{max} é melhor em termos de número de mensagens e mais apropriada para aplicações que não necessitam de um pequeno tempo de detecção. Por esta razão, η_{max} é mais apropriada para o monitoramento em redes de longa distância onde, em geral, as aplicações são mais tolerantes a atrasos. Como pode ser visto na Tabela 2, a estratégia η_{GCD} apresenta um menor tempo de detecção e melhor resultado para a métrica P_A. Portanto, sendo mais apropriada para o monitoramento em redes locais onde as aplicações geralmente não toleram períodos longos de atraso.

Tabela 2. Comparativo entre as duas estratégias: η_{max} and η_{GCD}.

	η (s)	T_D (ms)	T_M (ms)	T_{MR} (ms)	P_A	Num. de mensagens HB	Num. de falsas detecções
η_{max}	1.95	2458	1770	61190	0.9711	1754	2
η_{GCD}	1.00	1320	690	60592	0.9998	3551	2

5. Conclusão

Neste trabalho foi proposto o serviço QoS-CFDS para detecção de falhas de processos de sistemas distribuídos. O monitoramento dos estados dos processos é "auto-ajustável", executado com base nos parâmetros de QoS que permitem configurar o detector a partir dos requisitos simultâneos de múltiplas aplicações. Para calcular estes parâmetros de QoS foram propostas duas estratégias (η_{max} e η_{GCD}) que permitem deduzir o valor do intervalo de *heartbeat* (η) com base nas informações de QoS fornecidas para o QoS-CFDS. Neste sentido, observou-se que a estratégia η_{max} é mais apropriada para aplicações que toleram maiores atrasos na comunicação, ao passo que a estratégia η_{GCD} é indicada para casos em que as restrições forem maiores, obtendo maior probabilidade para uma resposta exata (P_A) e menor tempo para a detecção de uma falha. Os resultados experimentais mostram os benefícios dos ajustes realizados pelo QoS-CFDS com base nos valores de QoS, em especial a possibilidade de omitir, para as aplicações, falsas suspeitas cometidas pelo detector de falhas. Resultados para uso de CPU e memória demonstraram o serviço proposto é escalável em termos de número de objetos para monitoramento. Por fim, foi mostrada também a eficiência da estratégia utilizada para a adaptação do *timeout* em diferentes situações de atrasos de comunicação. Trabalhos futuros incluem a implementação de aplicações distribuídas tolerantes as falhas construídas sobre o serviço QoS-CFDS.

Referências

Bertier, M., Marin, O., and Sens, P. (2003). Performance Analysis of a Hierarchical Failure Detector. In *International Conference on Dependable Systems and Networks (DSN)*.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of ACM*, 43(2).

Chen, W., Toueg, S., and Aguilera, M. K. (2000). On the Quality of Service of Failure Detectors. In *International Conference on Dependable Systems and Networks (DSN)*.

de Sá, A. S. and de Araújo Macêdo, R. J. (2010). QoS Self-configuring Failure Detectors for Distributed Systems. In *International Conference on Distributed Applications and Interoperable Systems (DAIS)*. Springer-Verlag.

Dixit, M. and Casimiro, A. (2010). Adaptare-FD: A Dependability-Oriented Adaptive Failure Detector. *Symposium on Reliable Distributed Systems*.

Felber, P., Defago, X., Guerraoui, R., and Oser, P. (1999). Failure detectors as first class objects. In *International Symposium on Distributed Objects and Applications.*

Fetzer, C., Raynal, M., and Tronel, F. (2001). An adaptive failure detection protocol. In *Pacific Rim International Symposium on Dependable Computing*.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of ACM*, 32(2).

Greve, F., Sens, P., Arantes, L., and Simon, V. (2011). A Failure Detector for Wireless Networks with Unknown Membership. In *Euro-Par 2011 Parallel Processing*, volume 6853. Springer Berlin Heidelberg.

Harrington, D., Presuhn, R., and Wijnen, B. (2002). An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. Request for Comments 3411 (RFC3411).

Hayashibara, N., Cherif, A., and Katayama, T. (2002). Failure detectors for large-scale distributed systems. In *21st IEEE Symposium on Reliable Distributed Systems*.

Jacobson, V. (1988). Congestion Avoidance and Control. In *Symposium Proceedings on Communications Architectures and Protocols*.

Tomsic, A., Sens, P., Garcia, J., Arantes, L., and Sopena, J. (2015). 2W-FD: A Failure Detector Algorithm with QoS. In *2015 IEEE International Parallel and Distributed Processing Symposium, IPDPS*.

Turchetti, R. C. and Duarte Jr., E. P. (2015). Implementation of a Failure Detector Based on Network Function Virtualization. In *Workshop on Dependability Issues on SDN and NFV, International Conference on Dependable Systems and Networks (DSN)*.

Xiaohui, W. and Yan, Z. (2014). Adaptive failure detector A-FD. In *7th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*.

Xiong, N., Vasilakos, A., Wu, J., Yang, Y., Rindos, A., Zhou, Y., Song, W.-Z., and Pan, Y. (2012). A Self-tuning Failure Detection Scheme for Cloud Computing Service. In *IEEE 26th International Parallel Distributed Processing Symposium (IPDPS)*.

Timely hybrid synchronous virtual networks

Rasha Hasan[1], **Fernando Luís Dotti**[1]

[1]Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681 Prédio 32 sala 505, Porto Alegre, Brazil

rasha.hasan@acad.pucrs.br, fernando.dotti@pucrs.br

Abstract. *Network virtualization has been proposed in the last years, and it received special attention from both networking and distributed systems communities. However, specific applications' requirements, such as topology, security, and resilience, pose different challenges to the network embedding problem. Among these requirements lays the one of synchrony, that some applications demand time bounds for processing and communication. In this sense, Hybrid Synchrony Virtual Networks (HSVN) has been proposed to fulfill specific synchrony requirements and better support a considerable class of distributed systems. In our previous works, we gave attention to "space" HSVNs that was addressed to support hybrid synchrony systems in space. In this work, analogously, we discuss hybrid synchrony virtual networks for the "time" dimension. We regard the time HSVNs abstractions and techniques as being a refinement of the space HSVNs, since it further defines repeating time windows of synchrony while allowing the resource to behave asynchronously for a period of time. The timely hybrid model leads to the possibility of further sparing synchronous resources, if compared to the space model, as will be presented in this paper.*

1. Introduction

In the last three decades of research in Distributed Systems (DSs), one core aspect discussed is the one of synchrony. With an asynchronous system, we make no assumptions about process execution speeds and/or message delivery delays; with a synchronous system, we do make assumptions about these parameters [Schneider 1993]. In particular, in a synchronous system, the relative speeds of processes, as well the delays associated with communication channels are assumed to be bounded. The research on DSs has longly touched problems that base on the synchrony property of the system environment, for example the consensus problem [Dwork et al. 1988] where synchrony ensures progress of several distributed algorithms; another example is failure detection [Gartner 2001]. Many applications benefit from consensus and failure detection as building blocks in the distributed algorithms, for example Apache Cassandra [Hewitt 2011].

Synchrony in DSs impacts directly the complexity and functionality of fault-tolerant algorithms. Although a synchronous infrastructure contributes towards the development of simpler and reliable systems; yet such an infrastructure is too expensive and sometimes even not feasible to implement. On the other hand; a fully asynchronous infrastructure is more realistic, but some problems showed to be unsolvable in such an environment, such as the impossibility result by Fischer, Lynch and Paterson [Fischer et al. 1985]. The limitations in both fully synchronous or fully asynchronous

systems have led researchers to the development of partial synchronous distributed systems, which we call in our work, as well, by Hybrid synchronous Distributed Systems. Hybrid models assume intermediate levels of synchrony. In [Veríssimo 2006], Veríssimo presented the wormhole model, that considers the *space* dimension for hybrid synchrony, where one part of a system would behave synchronously, while another part would be fully asynchronous. Cristian and Fetzer proposed the timed-asynchronous model [Cristian and Fetzer 1999], where the system alternates between synchronous and asynchronous behavior over *time*. Due to the hybrid behavior, both models are stronger than asynchronous model and weaker than synchronous one.

During our research path [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015], we argued that Virtual Networks (VNs) and a suitable VN embedding process offer suitable environment for running distributed applications with partial synchrony. This has led to the abstraction of new type of virtual networks that we name *The Hybrid Synchrony Virtual Networks (HSVNs)*. They are virtual networks that have subsets of nodes and links that obey time bounds for processing and communication. Our previous contributions treated the *Space_HSVNs* considering physical resources hybrid in space, i.e. resources behave either synchronously or asynchronously during all the time. The *Space_HSVNs* are addressed to the DSs of hybrid synchrony in *space*, which we call *space hybrid synchronous systems*.

The timed-asynchronous model assumes that the system alternate between synchronous and asynchronous behavior. More specifically, according to [Dwork et al. 1988] partially synchronous systems can alternate between synchronous and asynchronous behavior, being hybrid in *time*. For each execution, there is a time after which the upper bound δ is respected by the system. This time is called *Global Stabilization Time (GST)*. Since the upper bound cannot hold forever, it is accepted that it holds just for a limited time Δ_s. In practical terms, Δ_s is the time needed for consensus to make progress or to be reached. We call these *timely hybrid synchronous systems*.

Actually, both models (i.e., hybrid models in *space* and in *time*) are not completely excludent. If a resource has no ability to behave synchronously, then it offers no time guarantees at all. However if a resource is able to behave synchronously, the space model defines that it is always synchronous while the time model defines that it behaves eventually synchronous as described above. From the point of view of resources utilization, we can regard the time model as being a refinement of the space model, since it further defines repeated windows of synchrony while allowing the resources to behave asynchronously for a period of time. It is possible to support timely hybrid synchronous systems with Space_HSVNs, adopting the assumption and embedding model proposed in [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015], but this choice would result in wasting synchronous resources because of reserving them for virtual demands that would behave synchronously only eventually (i.e., only during time windows). With the goal of sparing synchronous resources, we propose new type of HSVNs, that is the Time_HSVNs suitable for the timely hybrid synchronous nature of certain DSs.

The main contributions of this paper are: *(i)* define the assumptions and abstractions needed to characterize both Substrate Networks (SNs) and Virtual Networks (VNs) suitable for Time_HSVNs, *(ii)* and develop an embedding model for Time_HSVNs that answers the timely synchronous nature of the system and aware of sparing synchronous

resources which are relatively expensive. The rest of this paper will be organized as the following: Section 2 summarizes the related works; Section 3 details the abstractions and techniques we propose for the Time_HSVNs together with the embedding model; in Section 4 we evaluate the embedding model; and in Section 5 we conclude the work.

2. Related Work

The time-variant nature of networks has attracted considerable attention in the literature. Xie et al. [Xie et al. 2012] observed that during the networking intensive phases of applications, collision and competition occurs for the network resources, resulting in making the applications running time unpredictable. The uncertainty in execution time further translates into unpredictable cost, as tenants need to pay for the reserved virtual machines for the entire duration of their jobs. Xie et al. [Xie et al. 2012] propose the design of the first network abstraction (to the authors' best knowledge), TICV (Time Interleaved Virtual Clusters), that captures the time-varying nature of cloud applications, and they propose a systematic profiling-based methodology for making the abstraction practical and readily usable in today's data center networks. The network abstractions that [Xie et al. 2012] consider are similar to those proposed in [Guo et al. 2010, Ballani et al. 2011], but the last two works overlook the real time variant nature of resource requirements and simply assume that the customer will specify them somehow.

Zhang et al. conducted a series of research that considers the bandwidth (BW) variant nature of VNs during the process of resources provisioning, [Zhang et al. 2011, Zhang et al. 2012, Zhang et al. 2014]. The authors modeled the time-variant nature of the VNs demands as the combination of a *basic* sub-requirement, which exists all through the VNs lifetime, and a *variable* sub-requirement, which exists with a probability. For the basic sub-flows, fixed bandwidth is allocated (traditional BW sharing). But for the variable sub-flows, the authors consider specific design of the SN; they assume that the time is partitioned into frames of equal length, and each frame is further divided into slots of equal length. The authors develop two first-fit algorithms that map the variable sub-flows to the time slots on the SN. The first algorithm does not consider the inter-flows collision per slot, whereas the second algorithm is aware of it. The inter-flows collision per slot is calculated based on the probability of occurrence of the variable sub-flows.

In the topic of VNs mapping, we find that our work is near to the works concerned with delay constraints. For example, Zhang et al. [Zhang et al. 2010] propose a heuristic algorithm for mapping virtual multi-cast service-oriented networks subject to delay. Another work [Infuhr and Raidl 2011] addressed the VNs mapping problem with delay, routing and location constraints. Nevertheless, these works consider a homogeneous physical infrastructure and cannot be adopted for hybrid synchronous DSs that demand hybrid physical infrastructure. Moreover, these works do not consider applications with time-variant delay, both for processing speeds and messages communication.

Our work considers the synchrony time-variant nature of virtual networks, addressed for timely synchronous distributed systems. The problem raised in this paper (i.e., time_HSVNs embedding) could be solved using the models proposed in our previous works [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015] by overlooking the synchrony timely-variant nature of VNs, but this would result is reserving synchronous resources permanently for demands that require synchrony only during time windows. In

this paper, we argue that, adopting suitable abstractions and techniques, together with a suitable embedding model, increases the resources usage efficiency.

3. Time HSVNs: abstractions and techniques

Formerly in this paper, we referred to DSs that demand synchrony eventually during the system life, in other words, they demand synchrony during periods of time to allow the progress of these systems. We assume that this kind of systems are supported by virtual networks that reflect the timely synchrony nature. We name this type of VNs the *Time Hybrid Synchronous Virtual Networks* abbreviated to *Time-HSVNs*. In this section we detail about *(i)* the Time-HSVNs characterization; *ii)* the SN suitable design to answer the Time_HSVNs demands; and *(iii)* the Time-HSVNs embedding over the adopted SN.

3.1. Time-HSVNs characterization

Time-HSVNs carry the common features of typical VNs [Chowdhury and Boutaba 2010], but in addition, they need to be further characterized to allow them to reflect the timely synchrony nature. We consider that the synchrony demands of each VN has a cyclic pattern with the cycle T time units. During T, each virtual node and link demands synchrony once, for a certain period of time. The time windows when the virtual element is provided synchrony is named *the synchronous rounds*, and the time windows when it is not provided synchrony is named *the asynchronous rounds*. We assume that the client is able to define the synchronous rounds he needs within T, and he is able to express it to the virtual network provider. The client needs to be provided synchrony, eventually within T, during the specified time duration he expresses, without caring for *when* it will be provided within T. The VNs cyclic pattern makes them reflect the nature of timely synchronous DSs, which repeatedly demand eventual synchrony during the system life.

3.2. SN design

We assume the existence of certain mechanisms that guarantee building physical network elements (nodes and links) that behave synchronously. These mechanisms can be related to the type of physical materials used, or to the procedures followed for configuring them, such as admission control and Quality of Service policies. The exact mechanisms for building synchronous resources is out of the scope of our work, but we assume their existence. In our previous works with Space_HSVNs [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015], we distinguished between two types of resources; synchronous and asynchronous, where both types maintain their synchrony status during the system life. In the current step of our work, Time_HSVNs demand a refinement of the Space_HSVNs abstractions and techniques to suit better the new view of synchrony (i.e., periodic eventual synchrony).

In [Zhang et al. 2011], Zhang et al. propose a bandwidth sharing technique that allocates bandwidth (BW) in accordance with VNs traffic fluctuation as detailed in the related works. The authors consider specific design of the SN as the following: the time is partitioned into frames of equal length, and each frame is further divided into slots of equal length. The authors develop an algorithm that maps the variable sub-flows to the time slots on the SN in a way that the sub-flows maped to the same slot do not violate the BW capacity, neither exceed a collision threshold allowed, where the sub-flows collision

is calculated based on the probability of their occurrence. The proposed methodology allows an opportunistic bandwidth sharing between the sub-flows.

In our work, we inspire a suitable SN design from the work of Zhang et al. [Zhang et al. 2011] after adapting it in what matches our problem: *i)* in our work the virtual flows are of fixed BW demand during time (not opportunistic demands), thus, we disconsider collision probability; *ii)* the HSVNs demand synchrony once during T, see 3.1, so, we need only one time window during T that applies Zhang et al. technique. We name this time window *synchronous frame*. The synchronous frame is further partitioned into time slots of equal size, we name them *synchronous slots*; *iii)* the virtual demands mapped to a synchronous slot should not violate the physical BW capacity to eliminate competition and assure synchrony. The length of the synchronous frame and the number of time slots within a frame is related to the VNs number and demands. We assume that each virtual node and link do not demand synchrony slots that exceed the number of slots per synchronous frame.

3.3. Time_HSVNs Embedding

The virtualization architecture we adopt is the one proposed by Schaffrath et al. [Schaffrath et al. 2009]. We assume that the virtual network provider (VNP) has complete information about: *i)* the SN topology and its attributes (nodes Central Processing Unit (CPU), links bandwidth (BW), and synchronous slots number and length), and *ii)* the virtual networks topology and demands (nodes CPU, links BW, synchrony demands). The VNP receives the synchrony demands in term of time period, and translates it into number of synchronous slots of the SN slots. We deal with the case of off-line VNs embedding. The time_HSVNs will be provided synchrony during the synchronous frame. Out of the synchronous frame, additional asynchronous demands can be mapped and competition can occur. This does not pose any problem for demands that do not expect synchrony.

The Time_HSVNs embedding problem can be stated as the following: *How to map the virtual synchronous slots to the physical synchronous slots, with the objective of minimizing the mapping cost represented by the used BW?* The approach we followed for solving the Time_HSVNs was to benefit from our previous works on Space_HSVNs [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015], by refining the proposed model for Space_HSVNs to a new version that expresses the synchronous slots. Further, we enhanced the achieved solution to allow more VNs to be mapped on the same SN. So, the Time_HSVNs mapping would go through two phases as the following:

1. the macro mapping phase: This phase maps the virtual elements (nodes and links) to the physical resources that can support them. This phase leads to a mapping solution, that considers minimizing the physical bandwidth consumption. At the end of this phase, each virtual node and link will be mapped to a physical node and path that can support them. The macro mapping phase model is achieved by refining the Space_HSVNs mapping model to express synchronous slots.
2. the micro mapping phase: This phase increases the efficiency of the solution achieved in the macro mapping phase, by allowing embedding possible future VNs demands on the same given SN. this phase is performed individually for each physical node and link used in the macro mapping phase. The micro mapping phase maps the virtual synchronous demands to the physical synchronous

slots. For solving the micro mapping phase, we adopted an off-the-shelf problem from the literature due to its similarity, that is the Cutting Stock problem (CSP).

3.4. The Macro Mapping Phase

The inputs of this phase are: *(i)* the SN topology and attributes, and *(ii)* the VNs topologies and demands. And the output of this phase will be assigning each virtual node to one physical node and each virtual link to a physical path, where a path can be composed of one link or more. At this phase, the problem turns to be: *how to map the VNs on top of the SN with the least physical bandwidth consumption possible.* We formulate the macro mapping problem in the shape of a Integer Linear Program (ILP).

Variables definition - The SN is represented by an undirected graph $G(N, L)$, composed of a set of physical nodes N and links L. Analogously, each virtual network VN^k belonging to the set of virtual networks VN will be presented by an undirected graph $G^k(N^k, L^k)$. The number of synchronous slots provided by the physical node i and physical link (i, j) are $sync(i)$ and $sync(i, j)$. Analogously, $sync(i^k)$ and $sync(i^k, j^k)$ are the number of synchronous slots demanded by the virtual node i^k and link(i^k, j^k). Besides synchrony, two other attributes are considered for the SN and VN elements: nodes CPU, and links bandwidth (BW). The syntax for those attributes on the SN and VN respectively are: $cpu(i)$, $bw(i, j)$, $cpu(i^k)$, and $bw(i^k, j^k)$. Finally, we define the model output variables, they are: a binary function $\sigma(i^k, i)$ that expresses whether node $i \in N$ maps node $i^k \in N^k$, and a binary function $\rho(i^k, j^k, i, j)$ that expresses whether link (i, k) is part of the physical path that maps the virtual link (i^k, j^k). After solving the macro mapping model, each physical node i is mapping a set of virtual nodes $N^k(i)$, and each physical link (i, j) on the SN is mapping a set of virtual links $L^k(i, j)$.

The Macro mathematical model - It is formulated in the shape of an ILP as bellow:

Mapping objective- The Objective Function (1), we consider is inspired from [Hasan et al. 2013], which is to minimize the total bandwidth used.

Objective: minimize

$$\sum_{\forall VN^k \in VN} \sum_{\forall (i^k, j^k) \in L^k} \rho(i^k, j^k, i, j) \cdot sync(i^k, j^k) \cdot bw(i^k, j^k); \quad (1)$$

Mapping constraints-

- *Capacity constraints:*

for every $(i, j) \in L$ and every $(i^k, j^k) \in L^k$

$$\rho(i^k, j^k, i, j) \cdot bw(i^k, j^k) \leq bw(i, j) \quad (2)$$

for every $i \in N$ and every $i^k \in N^k$

$$\sigma(i^k, i) \cdot cpu(i^k) \leq cpu(i) \quad (3)$$

- *Nodes mapping constraints:*

for every $VN^k \in VN$, $i^k \in N^k$

$$\sum_{\forall i \in N} \sigma(i^k, i) = 1 \quad (4)$$

for every $VN^k \in VN$, $i \in N$

$$\sum_{\forall i^k \in N^k} \sigma(i^k, i) \leq 1 \tag{5}$$

- Links mapping constraint:
for every $VN^k \in VN$, $(i^k, j^k) \in L^k$, $i \in N$

$$\sum_{\forall j \in N} \rho(i^k, j^k, i, j) - \sum_{\forall j \in N} \rho(i^k, j^k, j, i) = \sigma(i^k, i) - \sigma(j^k, i) \tag{6}$$

- Nodes synchrony constraints:
for every $i \in N$

$$\sum_{\forall VN^k \in VN} \sum_{\forall (i^k) \in N^k} \sigma(i^k, i) \cdot sync(i^k) \leq sync(i) \tag{7}$$

- Links synchrony constraints:
for every $(i, j) \in L$

$$\sum_{\forall VN^k \in VN} \sum_{\forall (i^k, j^k) \in L^k} \rho(i^k, j^k, i, j) \cdot sync(i^k, j^k) \leq sync(i, j) \tag{8}$$

The capacity constraint (2) assures that the bandwidth of every virtual link does not exceed the bandwidth of the physical link mapping it. Similarly, constraint (3) is regarding nodes CPU. The node mapping constraint (4) assures that each virtual node is mapped once on a physical node. Constraint (5) assures that virtual nodes belonging to the same VN are not mapped on the same physical node. This is to achieve load balancing besides improving the reliability. This procedure minimizes the number of virtual nodes prone to failure by a physical node failure. For any virtual link (a, b), the links mapping constraint (6), adopted from [Zhang et al. 2010], assures the creation of a valid physical path. When mapping a set of virtual nodes $N^k(i)$ on one physical node i; the nodes synchrony constraint (7) assures that the number of virtual slots mapped on i do not exceed the number of synchrony slots provided by i. This constraint considers the worst case, when each virtual slot requires a complete synchrony slot alone without sharing. Similarly, constraint (8) represents the equivalent restriction regarding links synchrony.

3.5. The Micro mapping phase

This phase increases the efficiency of the solution achieved in the macro phase, by allowing embedding possible future VNs demands on the same SN. This is achieved by scheduling the synchronous demands efficiently within the synchronous frame. By viewing the problem at this stage as an optimization problem, the problem turns to be: *how to schedule the virtual demands within a synchronous frame minimizing the number of synchronous slots used*. Revising the literature, we found a very similar problem that is *the Cutting Stock Problem (CSP)* [Haessler and Sweeney 1991], which is one of the NP-hard problems cited by KARP [Karp 1972], and from the cutting stock problem we inspired the solution of the timely HSVNs micro mapping problem.

In operations research, the cutting-stock problem is the problem of cutting standard-sized pieces of stock material (e.g., paper rolls or sheet metal) into pieces of

specified sizes while minimizing material wasted. Translating the CSP elements into the micro mapping problem: the *specified sized pieces* are the synchronous slots demanded by the VNs; the *standard-sized pieces of stock material* is the synchronous frame on SN; *a pattern* is the set of demands accepted within a stock in the CSP, and it will be the set of virtual slots accepted within the physical slot in the micro mapping problem; and the objective of the CSP in to *minimizing the stock waste* and in our problem it would be to minimize the number of synchronous slots used within the synchronous frame.

The Micro mathematical model - We express it bellow for links but it goes similarly for nodes. Consider one physical link (i, j) with a synchronous frame of $sync(i, j)$ slots, that maps a set of virtual links $L^k(i, j)$, where every virtual link has two attributes: the number of synchronous slots demanded $sync(i^k, j^k)$ and the capacity of BW demanded $bw(i^k, j^k)$. Within each physical slot, the virtual synchronous slots that can be mapped to it form a pattern X_j [Haessler and Sweeney 1991]. The patterns are formed based on the BW of the virtual demands compared with the physical link BW. These patterns are the input to the micro model. The micro model aims at minimizing the number of synchronous slots used within the synchronous frame which is achieved by minimizing the total number of patterns (Equation (9)). Assuming a_{ij} is the number of times order i appears in pattern j [Haessler and Sweeney 1991]; constraint (10) assures providing every virtual link with a number of synchronous slots that is at minimum equal to the number of synchronous slots demanded $sync(i^k, j^k)$. The output of the micro mapping model will be telling which are the used patterns, and how many of each pattern is needed.

Objective: minimize

$$\sum_{\forall j \in PATTERNS}(X_j) \tag{9}$$

Cutting constraint
for every $i \in L^k(i, j)$

$$\sum_{\forall j \in PATTERNS} (a_{ij} \cdot X_j) \geq sync(i^k, j^k) \tag{10}$$

After performing the micro mapping phase, the SN is updated (used BW and CPU is substracted from the SN nodes and links capacity), and the macro mapping phase can run again, allowing more VNs to be mapped on the same SN. The combination on the macro phase, the micro phase, and the SN updating we name *an optimization cycle* (Opt_cyc). Figure 1 illustrates two optimization cycles for mapping groups of virtual links on one physical link. The physical link updating happens either by reducing its capacity, or reducing the number of synchronous slots it supports. Either way, there will be a waste in the physical bandwidth, we refer to the bandwidth waste resulted by reducing the physical capacity W_h, and by reducing the synchronous slots W_v. The updating approach chosen is the one with the least waste (the smaller value between W_h and W_v is marked with a star * in the figure). This updating methodology we follow is because the macro mapping phase (at the beginning of each optimization cycle) considers physical slots of equal capacity and fully empty.

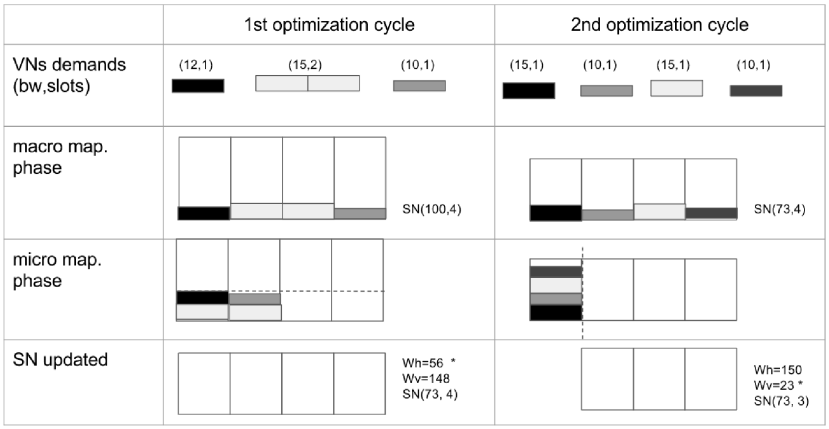


Figure 1. An illustrative scheme for the HSVNs optimization cycles

4. Performance Evaluation

The Time_HSVNs abstractions we considered together with the Time_HSVNs embedding model are supposed to lead to further sparing of synchronous resources, if compared to the space model. We run preliminary experiments that allow investigating the performance of the proposed embedding approach of Time_HSVNs. The aspects considered during the analysis of our model are: *(i)* the embedding cost; *(ii)* the physical resources load; *(iii)* the optimization time; *(iv)* the topology of the physical subnetwork composed of the used resources; and *(v)* the micro mapping phase efficiency.

Experiments were designed as a full factorial [Jain 1991], exploring all possible combinations between the networks parameters. Such choice of experiments was done by other works like [Bays et al. 2012]. Similar to [Yu et al. 2008, Bays et al. 2012], physical and virtual networks were randomly generated. For this we used BRITE tool (Boston university Representative Internet Topology gEnerator) [Medina et al.] with Waxman model [Waxman 1988]. We implemented the mathematical model with ZIMPL language (Zuse Institute Mathematical Programming Language) [Koch 2004], both for the macro and micro mapping phases, and we used CPLEX [IBM] to solve the Integer Program (IP), running on a computer Intel HM75, Core i3-3217U 1.80 GHz (Giga Hertz), cash 3 MB (Mega Byte), Random Access Memory (RAM) of 2 GB (Giga Byte), DDR3 and operating system Xubuntu 14.04.

In all the following experiments, the substrate network size was fixed to 15 nodes. Initially all CPUs (Central Processing Unit) of SN nodes are free, and links Band Width (BW) is uniformly distributed between 1-3 Gbps (Giga bit per second). We ran twelve experiments divided into three groups, A, B and C, with VNs total size of 10, 20, and 30 nodes in each group respectively. The VNs were generated with 3, 4, or 5 nodes each, and CPU demands 10%, 15%, or 25% of the SN nodes CPU capacity, and BW demands uniformly distributed between 100 Mbps (Mega bit per second) and 1 Gbps. VN nodes demand one synchronous slot per T. In scenarios 1, 2, 3 and 4 of each group, the virtual links synchronous slots demanded varies between 1, 2 , 3 , and 4 slots. The SN provides periodically, each $T = 20$ seconds, a synchronous frame of 4 seconds length, divided into four equal time slots. Table 1 details the experiments parameters.

The first parameter evaluated is the mapping cost represented by the used BW,

Table 1. Experiments parameters

Expe.	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
VN size	10 nodes				20 nodes				30 nodes			
Virtual links sync. slots	1	2	3	4	1	2	3	4	1	2	3	4
Virtual nodes sync.	1 slot per T											
each VN size	3,4,5 nodes											
VNs BW	uniformly distributed: 100Mbps-1Gbps											
VNs CPU	10,15,25 % of SN nodes CPU											
SN size	15 nodes											
SN BW	uniformly distributed: 1 Gbps-3 Gbps											
SN CPU	nodes fully free initially											

which is the model objective function. We evaluate our results by comparing them with the BW used in case the experiments in Table 1 were mapped using the space model and the SN previously addressed in [Hasan et al. 2013], see Figure 2. With the time model, the used BW indicated is the one consumed within the synchrony frame. To allow just comparison, we calculate the BW used in the space model during time window equal to the frame length. We note down the following main observations:*(i)* within one experiments group, the used BW is proportional to the number of synchronous slots demanded by the VNs. For example, in scenario A2 the BW used is 13.624 Mbps/frame, and it is 20.436 in scenario A3, the proportion between the two figures is the proportion of the number of synchronous slots demanded in each scenario 2/3. *(ii)* by comparing the counterparts experiments of the three groups (e.g., A2, B2, and C2) we notice that the BW used increases. This is due to the VNs size increment, which tends naturally to reserve more resources. *(iii)* within each experiment's group, the BW used with the space model is equal to the maximum BW used within the group (i.e., experiment 4 of each group). *(iv)* the time model is more efficient as it spares more resources. For example, mapping the VNs in scenario B3 with the time model spares 33,33% of the resources needed when mapping the same set of VNs with the space model. And the spared ratio increases when the synchronous demands within T decreases, e.g. in B2, the time model spares 100% of the resources, and in B1 spares 300%.

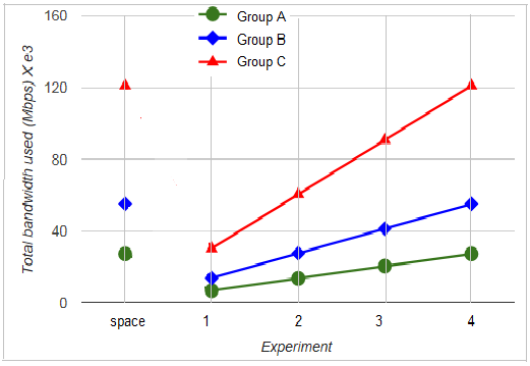


Figure 2. Used bandwidth

The second parameter evaluated is the physical resources load. This study is useful when done on a scenario that can possibly load the SN. We chose scenario C2 (VNs of big size). Figure 3 depicts the Cumulative Distributed Function (CDF) for physical nodes and links in experiment C2. We note that 80% of the SN nodes had a load that varies between 10% and 60%, only 13.33% had a load between 60% and 80%, and no nodes were highly loaded more than 80%. And regarding the physical links, we note that, 41.37% of the SN

links had a load that ranges between 10% and 60%, and only 14% of the physical links had a load that exceeded 60%, and no links were fully loaded. So, the SN resources seem to have load distribution which is good, since concentrating the load in certain elements will result in congestion, leading to block mapping certain VNs in the future. This is achieved because, the proposed model does not push the mapping process to exhaust the used physical resources before allocating new ones, rather, all resources are given the same chance to be chosen, as long as they allow mapping on the shortest path.

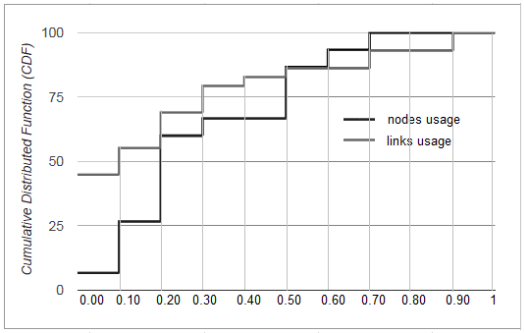


Figure 3. CDF for resource usage in experiment C2

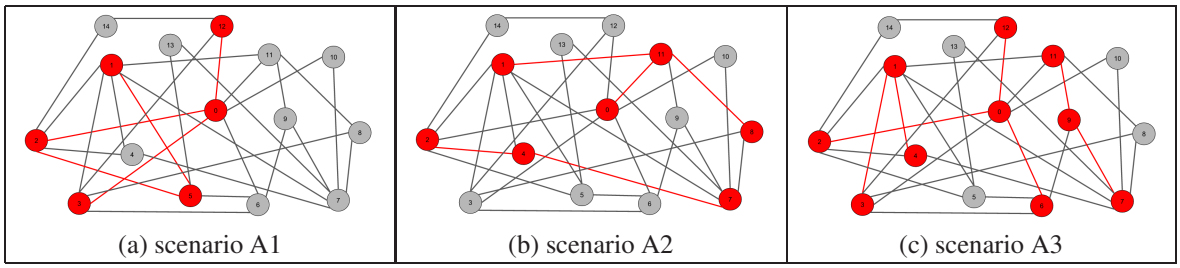
The third parameter evaluated is the mapping time. The optimization process reached its end with scenarios A and B. Whereas in scenarios C, the optimization was terminated with optimization gap less than 2%. We took this decision when the optimization progressed slowly without much gain. For example, in scenario C3, it took 30 minutes to reach a solution with 4.46% gap, then another 33 minutes to reach another solution with 1.86% gap. In realistic scenarios, the client might prefer a semi-optimal solution in a short computational time, than an optimal solution after long time. From Table 2, we notice that, *(i)* most of the scenarios demanded optimization time that is less than 20 minutes, which is an acceptable computational time; *(ii)* we notice that the time model demands more time than the space model. And we notice that the difference between both increases with the increment of the problem size (i.e., VNs size and synchronous slots demanded) which increases the number of variables that need to be solved by the optimization process.

Table 2. Embedding time (in minutes)

Group.	space	exp.1	exp.2	exp.3	exp. 4
A	0.13	0.07	0.08	0.09	0.19
B	0.75	1.46	1.16	5.31	18.13
C	8	15.09	46.55	65.95	38.89

The fourth parameter studied is the topology of the physical subnetwork composed of the physical used elements (i.e., nodes and links). Figure 4 illustrates this topology for scenarios A1, A2, and A3. The topologies under study are the ones in red. We notice that, even though all these scenarios with the same VNs size (10 nodes), yet the model tends towards reserving more physical elements with the increment of the synchronous slots demanded by the VNs. For example, in scenario A1 the physical subnetwork under study is composed of 6 nodes, whereas in scenario A3 it is composed of 10 nodes. Previously, we noted down that the model tends towards distributing the BW load on the physical

resources, Figure 3. Now we add that the model tends also towards distributing the synchrony load as well. So, when the VNs increase their synchrony demands (i.e., number of synchrony slots), the model tends towards reserving new elements. This behavior avoids congesting the synchronous frames of the used elements, allowing mapping new arriving VNs. Because the time HSVNs will be blocked or by exhausting the SN CPU and BW, or by exhausting the SN synchronous slots.



(a) scenario A1 (b) scenario A2 (c) scenario A3

Figure 4. Topology divergence of used physical resources

The fifth parameter studied is the micro mapping model efficiency represented by the number of VNs accepted. We run this study on the case of one physical link and an endless queue of virtual links attended in order. We consider five experiment groups with different load range, Table 3. We run three experiments per group, with different synchrony demands. In each experiment several opt_cyc are performed till the physical link is exhausted and no demands are accepted. Our main observations: *i)* the efficiency decreases when the virtual links load increases. For example, the efficiency in group K was 9, 13, and 16 whereas in group L it was 6, 8, and 12. *ii)* the efficiency increases when the maximum number of synchronous slots demanded decreases. For example, in group K, when the maximum number of synchronous slots demanded decreased from 3 to 1, the model efficiency increased from 9 to 16; *iii)* the micro model efficiency is the same in group N and O, the reason is that, both groups are with high virtual links load, this does not allow slots sharing between virtual demands, and the mapping solution achieved in the macro mapping phase cannot be optimized further with the micro mapping phase.

Table 3. Number of mapped virtual links with different load and synchrony demands

Scenario	VNs load/SN capacity	$sync(i^k, j^k)$=1,2,3 slots	$sync(i^k, j^k)$=1,2 slots	$sync(i^k, j^k)$=1 slot
K	(0-20] %	9	13	16
L	(20-40] %	6	8	12
M	(40-60] %	5	6	8
N	(60-80] %	1	2	4
O	(80-100] %	1	2	4

5. Conclusion

In our previous works, we gave attention to *space* HSVNs that was addressed to support hybrid synchrony systems in space. In this work, analogously, we discuss hybrid synchrony virtual networks for the *time* dimension. The main contributions of this paper are: *i)* characterize the time_HSVNs to reflect the synchrony time-variant nature; *ii)* adopt a suitable design for the SN to support the demands; and *iii)* propose a resources allocation model. Simulation results show that the proposed embedding framework answers the synchrony time-variant demands efficiently (spares synchronous resources), distributes

the load over the physical resources (nodes and links), and has an acceptable computation time for reaching the embedding solution. In addition, topological study of the subnetworks composed of the used resources on the SN showed that, the embedding model is aware of the synchronous demands variation, and the resulting subnetwork scatters more on top of the SN when the synchronous demands increase. Further study of the micro-mapping phase showed that its efficiency is a function of the VNs load and synchrony.

At this phase of our work, we consider that the synchrony pattern of each virtual node and link is independent of each other, in future works, we intend to consider cases when VNs elements synchrony is mutually dependent, to reflect better the nature of the time hybrid synchronous DSs.

Acknowledgement

This work was supported in part by CAPES PVE - Grant No. 88881.062190/2014-01.

References

Ballani, H., Karagiannis, T., and Rowstron, A. I. T. (2011). Towards predictable datacenter networks. In *Proc. of the 11th ACM SIGCOMM conference*, pages 242–253.

Bays, L. R., Oliveira, R. R., Buriol, L. S., Barcellos, M. P., and Gaspary, L. P. (2012). Security-aware optimal resource allocation for virtual network embedding. In *the 8th International Conference on Network and Service Management (CNSM)*.

Chowdhury, N. M. and Boutaba, R. (2010). A survey of network virtualization. *Computer Networks*, 54(5):862–876.

Cristian, F. and Fetzer, C. (1999). The timed asynchronous distributed system model. *IEEE Trans. on Parallel and Distributed Systems*, 10(6):642–657.

De Oliveira, R. R., Dotti, F. B., and Hasan, R. (2015). Heuristicas para mapeamento deredes virtuais de sincronia hibrida. In *33o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 291–304.

Dwork, C., Lynch, N., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382.

Gartner, F. C. (2001). A gentle introduction to failure detectors and related subjects. Technical report, Darmstadt University of Technology, Department of Computer Science.

Guo, C., Lu, G., Wang, H. J., Yang, S., and Kong, C. (2010). Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *ACM CoNEXT*.

Haessler, R. W. and Sweeney, P. E. (1991). Cutting stock problems and solution procedures. *European Journal of Operational Research*, 54:141–150.

Hasan, R., Mendizabal, O. M., and Dotti, F. L. (2013). Hybrid synchrony virtual networks: Definition and embedding. In *The Thirteenth International Conf. on Networks*.

Hasan, R., Mendizabal, O. M., Oliveira, R. R. D., and Dotti, F. L. (2014). A study on substrate network synchrony demands to support hybrid synchrony virtual networks. In *32o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Hewitt, E. (2011). *Cassandra: The Definitive Guide*. OReilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

IBM. Cplex. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer.

Infuhr, J. and Raidl, G. R. (2011). Introducing the virtual network mapping problem with delay, routing and location constraints. In *Proc. of 5th International Networking Optimization Conference (INOC)*.

Jain, R. (1991). *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling*, chapter 16. Wiley Computer Publishing.

Karp, R. M. (1972). *Complexity of Computer Computations*, chapter Reducibility Among Combinatorial Problems, pages 85–103. Springer.

Koch, T. (2004). *Rapid mathematical programming*. PhD thesis, Tichnische Universität Berlin.

Medina, A., Lakhina, A., Matta, I., and Byers, J. Brite: Boston university representative internet topology generator. In *Available online: http://www.cs.bu.edu/brite*.

Schaffrath, G., Werle, C., Papadimitriou, P., Feldmann, A., Bless, R., Greenhalgh, A., Wundsam, A., Kind, M., and Maennel, O. (2009). Network virtualization architecture: Proposal and initial prototype. In *Proc. of 1st ACM workshop on virtualized infrastructre systems and architectures, ser. VISA'09*, pages 63–72, New York, NY, USA.

Schneider, F. B. (1993). *Distributed systems (2nd Ed.)*, chapter 2 (What Good are Models and What Models are Good?). Wesley Publishing Co.

Veríssimo, P. E. (2006). Travelling through wormholes: a new look at distributed systems models. *ACM SIGACT News*, 37(1):66–81.

Waxman, B. M. (1988). Routing of multipoint connections. *Selected Areas in Communications*, 6(9).

Xie, D., Ding, N., and Hu, Y.C. andKompella, R. (2012). The only constant is change: Incorporating time-varying network reservations in data centers. In *SIGCOMM 12*.

Yu, M., Yi, Y., Rexford, J., and Chiang, M. (2008). Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM-SIGCOMM*, 38(2).

Zhang, M., Wu, C., Jiang, M., and Yang, Q. (2010). Mapping multicast service-oriented virtual networks with delay and delay variation constraints. In *in Proc. to IEEE Global Telecommunication Conference (GLOBECOM)*, Miami, FL.

Zhang, S., Qian, Z., Tang, B., Wu, J., and Lu, S. (2011). Opportunistic bandwidth sharing for virtual network mapping. In *Global Telecommunications Conference*, pages 1–5.

Zhang, S., Qian, Z., Wu, J., and Lu, S. (2012). An opportunistic resource sharing and topology-aware mapping framework for virtual networks. In *Proc. to IEEE INFOCOM conference*, pages 2408–2416.

Zhang, S., Qian, Z., Wu, J., Lu, S., and Epstein, L. (2014). Virtuail network embedding with opportunistic resource sharing. *IEEE Journal of Transactions on Parallel and Distributed Systems*, 25(3):816–827.

Uma Estratégia Paralela e Autônoma para a Projeção de Modelos de Nicho Ecológico em Ambientes Computacionais Heterogêneos

Fernanda G. O. Passos, Vinod E. F. Rebello

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói, RJ – Brasil.

{fernanda,vinod}@ic.uff.br

Abstract. *Ecological Niche Modelling (ENM) is an important process to help ecologists understand and predict the potential geographic distribution of species. In addition to creating correlative models for each species, the projection of the model onto a geographical environment is an essential step in the process. Given the demand for improved precision and the need to address wider geospatial domains, using larger data sets means that the adopted methods incur relatively higher processing, memory and I/O demands. This paper proposes a new parallel and autonomic algorithm for the projection stage of an existing ENM tool widely used in large heterogeneous computing environments. The proposal is compared to the current default sequential implementation and a parallel MPI version both distributed with the software tool. An empirical analysis reveals gains of over 170% in terms of performance against the original MPI version, while the experiments also indicate improved scalability with efficiencies above 80%, in evaluations with up to 128 processors.*

Resumo. *A modelagem de nichos ecológicos (Ecological Niche Modelling – ENM) é um processo importante para ajudar ecologistas a entender e prever a distribuição geográfica potencial de espécies. Junto à criação de modelos correlativos para cada espécie, a projeção do modelo em um ambiente geográfico é um passo essencial neste processo. Devido à necessidade para melhorar a precisão e para tratar grandes domínios geoespaciais, tal método pode demandar níveis de processamento, memória e E/S, relativamente altos. Este artigo propõe um novo algoritmo paralelo e autônomo para a etapa de projeção de uma ferramenta ENM existente bastante utilizada em ambientes computacionais heterogêneos de larga escala. A proposta é comparada com a implementação sequencial padrão e uma versão paralela em MPI, ambas disponibilizadas com a ferramenta. Uma análise empírica indica ganhos em termos de desempenho acima de 170% em relação a versão MPI assim como uma melhor escalabilidade com eficiências acima de 80%, em testes com até 128 processadores.*

1. Introdução

A modelagem de nichos ecológicos – ENM (*Ecological Niche Modelling*) – é um procedimento comum para determinar a extensão da distribuição geográfica das espécies, provendo um procedimento poderoso para prever a potencial distribuição de espécies em diferentes contextos geográficos e temporais, assim como para estudar outros aspectos da

evolução e ecologia. ENM tem sido bastante usado em várias situações como por exemplo: busca de espécies raras ou ameaçadas de extinção; identificação de regiões adequadas para a (re-)introdução de espécies; previsão do impacto das mudanças climáticas sobre a biodiversidade; definição e avaliação de áreas protegidas; prevenção da propagação de espécies invasoras, entre outras aplicações importantes.

Aplicações de ENM combinam informações sobre a ocorrência de espécies (bióticas) com bases de dados ambientais (abióticas) na forma de camadas rasterizadas geo-referenciadas (tais como temperatura, precipitação e salinidade) para gerar potenciais modelos de distribuição. Este processo inclui uma combinação de etapas dependentes (em *workflow*) como: a criação, teste e projeção de modelos. Os modelos são gerados por algoritmos geralmente estatísticos como máxima entropia ou de inteligência artificial como redes neurais artificiais [Muñoz et al. 2011].

Este artigo irá tratar a paralelização da etapa de projeção em domínio geoespacial de modelos de nichos ecológicos, uma vez que o processamento deste tipo de dados é custoso. Tanto um algoritmo sequencial quanto um paralelo e distribuído é disponibilizado em uma ferramenta existente chamada OpenModeller [Muñoz et al. 2011]. Além disso, atualmente, existem projetos como [Amaral et al. 2015, Leidenberger et al. 2015, EUBrazil Cloud Connect 2015], que se baseiam no openModeller para prover serviços em nuvem computacional de ENM. No entanto, tais implementações não levam em consideração as características de recursos heterogêneos e compartilhados de ambientes como nuvens. O objetivo deste trabalho é propor um novo algoritmo paralelo e distribuído para a projeção ENM de modo que ele seja capaz de autogerenciar sua execução para torná-la mais eficiente. Para isto, este algoritmo é executado junto a um sistema gerenciador de aplicações chamada EasyGrid AMS.

Inicialmente, na Seção 2 são relatados trabalhos que fazem uso da ferramenta OpenModeller e trabalhos que exploram a paralelização de problemas que tratam dados geográficos. A Seção 3 descreve o caso de estudo deste artigo: a projeção do modelo de nicho ecológico, assim como sua implementação sequencial na ferramenta OpenModeller. A sua versão paralela MPI é explicada na Seção 4. Na Seção 5, o EasyGrid AMS e seu modelo de programação são apresentados para que, na Seção 6, a proposta do novo algoritmo paralelo e autônomo da projeção ENM seja exposta. A Seção 7 defende a proposta através de experimentos e resultados que demonstram sua eficiência de desempenho em relação a outras propostas. Por último, algumas conclusões sobre o trabalho são apresentadas.

2. Trabalhos Relacionados e openModeller

Existem dois pontos importantes que podem ser abordados como trabalhos relacionados neste artigo: 1) o tratamento de dados geoespaciais em paralelo e 2) o uso do *framework* ENM chamado OpenModeller, que, por sua vez, trata dados geográficos.

O estudo de aplicações que tratam dados geoespaciais ganharam força nos últimos anos devido à propagação de ferramentas como *Sistemas de Informação Geográfico* (GIS – *Geographic Information System*) [Bonham-Carter 2014] capazes de permitir a visualização e análise simples dos dados espacial e suas informações através de mapas, facilitando o trabalho dos especialistas. Por muitas vezes as aplicações usarem intensivamente uma grande quantidade de dados, existem trabalhos que exploram a paralelização

delas. Em [Abrahart and See 2014, Aji et al. 2013], diversos trabalhos são descritos considerando a implementação da distribuição de blocos estáticos de dados entre processadores, mas questionando a falta de modelos de programação para ambientes mais complexos. Os trabalhos em [Zhao et al. 2013, Aji et al. 2013] apresentam implementações para GPU seguindo mesma distribuição de dados dos anteriores.

2.1. O framework openModeller

O openModeller (OM) é uma ferramenta para ENM desenvolvida pelo Centro de Referência em Informação Ambiental (CRIA), junto com outros parceiros nacionais e internacionais [Sutton et al. 2007, Muñoz et al. 2011], bastante difundida entre a comunidade biológica e ecológica [Geller and Melton 2008, Ramachandra et al. 2010, Zarco-González et al. 2013]. Ele trabalha com modelos de distribuição potencial e inclui mecanismos para a leitura de dados ambientais e de ocorrência de espécies, seleção de camadas ambientais sobre as quais o modelo deve se basear, criação de um modelo de nicho fundamental, projeção de modelo em um cenário ambiental e escrita de dados em vários tipos de arquivo relativos a grandes imagens.

Vários algoritmos são providos na forma de *plugins* para a criação de modelos [Muñoz et al. 2011], incluindo:

- BIOCLIM (*Bioclimatic envelopes*): envoltórios bioclimáticos;
- CSMBS (*Climate Space Model*): modelo de clima espacial;
- GARP (*Genetic Algorithm for Rule-set Production*): algoritmo genético para produção de conjunto de regras;
- GARP_BS (*Best Subsets*): GARP com melhores subconjuntos;
- ENFA (*Ecological-Niche Factor Analysis*): análise de fator do nicho ecológico;
- ENVSCORE (*Envelope Score*): pontuação de envoltório;
- ENVDIST (*Environmental Distance*): distância ambiental;
- MAXENT (*Maximum Entropy*): entropia máxima;
- NICHE_MOSAIC (*Niche Mosaic*): mosaico de nicho;
- ANN (*Artificial Neural Network*): rede neural artificial;
- RF (*Random Forests*): florestas aleatórias;
- SVM (*Support Vector Machines*): máquina de vetor de suporte, e;
- AQUAMAPS: algoritmo para organismos aquáticos.

3. Estudo de Caso: Projeção do Modelo de Nicho Ecológico

Em uma modelagem de nicho ecológico, uma etapa muito importante e que requer grande processamento de dados geoespaciais é a projeção do modelo. Uma vez que o modelo de distribuição seja gerado, a projeção no mapa pode ser realizada. Tais modelos podem ser projetados em diferentes regiões geográficas, em diferentes cenários ambientais, tornando-se possível prever o impacto das mudanças climáticas sobre a biodiversidade, evitar a propagação de espécies invasoras, identificar aspectos geográficos e ecológicos da transmissão de doença, ajudar no planejamento de conservação, guiar campos de pesquisas, entre muitos outros usos [Peterson et al. 2011].

OM project – openModeller *project* – é a ferramenta do openModeller capaz de fazer a projeção dos modelos de distribuição. Ele irá gerar uma imagem com coordenadas (x, y) em uma determinada região geográfica cujos valores associados a cada ponto representam a probabilidade de se encontrar a espécie de acordo com os dados ambientais.

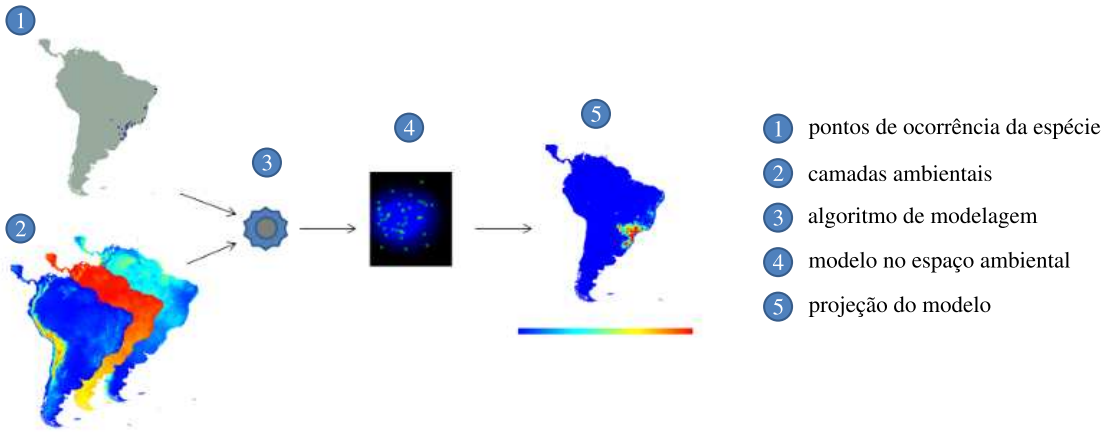


Figura 1. Exemplo de modelagem e projeção de um nicho ecológico [OpenModeller Team 2015].

A Figura 1 representa a modelagem e projeção de um nicho ecológico, onde o item 5 mostra o resultado da projeção de um modelo. A escala de cores que varia de azul a vermelho indica a adequação da espécie na região (vermelho é alta e azul é baixa). O usuário seleciona camadas de dados ambientais para a projeção e pode, também, selecionar uma máscara geoespacial, para projetar o modelo em uma área diferente de onde foi gerado. A imagem gerada geralmente é de alta resolução de *pixels* que deve ser visualizada por programas específicos como os sistemas de informação geográfica Quantum GIS ou QGIS, de código aberto [QGIS Project 2015].

Resumidamente, o Algoritmo 1 apresenta a implementação sequencial da projeção de um modelo de nicho ecológico e executa os seguintes passos para cada ponto (x, y):

1. adquire todos os dados das camadas ambientais no ponto (x, y) (corresponde à Linha 3 do algoritmo);
2. aplica o modelo no ponto (x, y) considerando tais dados ambientais e, então, gera uma probabilidade P (corresponde à Linha 4 do algoritmo);
3. converte P em valor da imagem e o escreve no arquivo na posição (x, y) (corresponde à Linha 5 do algoritmo).

As entradas do algoritmo são o modelo e os dados ambientais e, a saída, um arquivo contendo o mapa com as probabilidades. O modelo é previamente gerado pela etapa de modelagem através de um algoritmo descrito na Seção 2.1. Assim, a ferramenta do OpenModeller *OM project* tipicamente faz a leitura, processamento e escrita de uma grande quantidade de dados geográficos. Isto faz com que haja a iniciativa da criação de versões paralelas. Atualmente, incluída na ferramenta, existe um algoritmo paralelo desenvolvido para *clusters* de computadores homogêneos.

4. Versão MPI Original

A paralelização original do *OM project* usa um modelo semelhante ao mestre-trabalhador. Além dos processos trabalhadores que computam a projeção, existem dois processos mestres (ao invés de um) responsáveis por distribuir e receber as soluções, respectivamente, o que difere do modelo mestre-trabalhador tradicional, mas que tem efeito semelhante. Denomina-se *mestreD* o processo que distribui os intervalos de dados para cada processo

Algoritmo 1: Algoritmo sequencial de projeção do modelo de um nicho.

Entrada: $modelo$ - modelo previamente gerado pela ferramenta OM.
$dados\_ambientais$ - dados ambientais.
Saída: $mapa$ - arquivo que contém a projeção.

```
1 para x ← 0 a Dim_x − 1 faça
2     para y ← 0 a Dim_y − 1 faça
3         amb ← dados_ambientais na posição (x, y)
4         aplica modelo a amb e coloca o valor em Pr
5         escreve Pr em mapa na posição (x, y)
```

trabalhador e *mestreR* o processo que recebe as soluções de cada processo trabalhador e as escreve no arquivo imagem de saída (que contém a projeção).

O algoritmo de *mestreD* divide o número total de pontos a serem projetados (pontos da imagem) em blocos (valor padrão é 30.000) e os distribui aos processos trabalhadores sob demanda. Nesta aplicação, um bloco corresponde a um conjunto de linhas do mapa a ser projetado e, por isto, cada bloco é associado a (l_i, l_j), onde l_i é a linha inicial e l_j é a linha final. Para isso, um pedido de requisição deve ser feito pelos processos trabalhadores ao processo *mestreD* para que ele envie os dados informando qual o bloco a ser computado. Uma vez calculado, este bloco é enviado para o processo *mestreR* que irá receber os dados e escrevê-los no arquivo de saída. Quando todos os blocos forem calculados e escritos no arquivo pelo processo *mestreR*, o algoritmo termina. A separação de atividades do mestre entre 2 processos, *mestreD* e *mestreR*, é feita para que a distribuição de tarefas não seja sobrecarregada pela recepção dos dados, que é mais custosa.

Algoritmo 2: Algoritmo usado pelos trabalhadores da versão MPI original.

Entrada: $modelo$ - modelo previamente gerado por um algoritmo.
$dados\_ambientais$ - dados ambientais.
Dim_y - dimensão Y do mapa completo.

```
1  fim ← FALSO
2  enquanto fim = FALSO faça
3      Envia requisição a mestreD
4      Recebe (l_i, l_j) de mestreD
5      se l_i = −1 então
6          fim ← VERDADEIRO
7      senão
8          pacote ← ∅
9          para x ← l_i a l_j − 1 faça
10             para y ← 0 a Dim_y − 1 faça
11                 amb ← dados_ambientais na posição (x, y)
12                 Aplicar modelo em amb e colocar o valor em Pr
13                 pacote ← pacote ∪ (x, y, Pr)
14         Envia pacote para mestreR
```

As etapas de execução dos processos trabalhadores da versão original são apresentadas pelo Algoritmo 2. Neste caso, as entradas são o modelo, previamente gerado por um algoritmo na fase de modelagem, os dados ambientais e a dimensão Dim_y que denota o tamanho de uma linha. A variável fim controla o laço de repetição iniciado na linha 2. O término da repetição é feito nas linhas 5 e 6, onde é verificado se o l_i possui o valor -1 (bloco nulo). Caso positivo, a variável fim é alterada para verdadeiro e, então o laço de repetição termina. No início do laço, nas linhas 3 e 4, ocorre o envio da requisição para o processo $mestreD$ e o recebimento do bloco requerido em (l_i, l_j). Se forem valores válidos, o cálculo da projeção é iniciado. Para tal, um conjunto chamado $pacote$ é inicializado por $\varnothing$, na linha 8, e nele serão armazenados os valores já projetados. As linhas 9 e 10 indicam dois laços aninhados onde acontece o percorrimento do bloco atribuído à tarefa. Para cada ponto (x, y) deste bloco, o modelo é aplicado nos dados ambientais, a probabilidade é gerada e este resultado é guardado no conjunto $pacote$ (linha 13). Na linha 14, o envio do pacote completo para o $mestreR$ é realizado.

5. EasyGrid AMS e seu Modelo de Programação

O EasyGrid AMS [Boeres and Rebello 2004] é um sistema gerenciador de aplicações (AMS) que funciona como um meio entre a aplicação e o sistema computacional, de forma a facilitar o uso dos recursos considerando características da própria aplicação. O *middleware* EasyGrid AMS possui um gerenciamento hierárquico de três níveis e sua filosofia [Oliveira and Rebello 2011] indica que as aplicações devem ser gerenciadas por um sistema simples agregado a seu próprio *middleware* autônomo.

Como características autônomas, o EasyGrid AMS possui um escalonador de tarefas estático e dinâmico [Nascimento et al. 2007a] que fornecem auto-otimização (*self-optimization*) às tarefas da aplicação; um mecanismo tolerante a falhas por reexecução de tarefas e *checkpoint* no nível do gerenciador global que é capaz de detectar e recuperar as falhas da aplicação (autorrecuperação ou *self-healing*) [Silva and Rebello 2007]. A autoconfiguração (*self-configuring*) pode ser implementada de acordo com características da aplicação e, portanto, é específica por aplicação ou classe de aplicações. Trabalhos da literatura mostram bons resultados utilizando o EasyGrid AMS [Nascimento et al. 2007b, Silva and Rebello 2007, Ribeiro et al. 2010, Oliveira and Rebello 2010].

O seu modelo de programação é baseado no 1Ptask [Nascimento et al. 2007b], um modelo diferente do tradicional aqui chamado 1Pproc. O modelo 1Pproc considera um processo por processador e cada processo adquire toda a carga de trabalho atribuída a ele. Já o modelo 1Ptask indica um processo por tarefa. Assim, cada processo representa uma tarefa que geralmente é uma parte pequena de todo o trabalho da aplicação, tendo, portanto, um quantidade grande de processos por processador.

6. Versão Autônoma com EasyGrid AMS

A versão MPI original apresenta basicamente 2 problemas: 1) o algoritmo é centralizado e a troca de mensagens pelos mestres torna um gargalo de execução, prejudicando o desempenho. 2) o modelo de programação 1Pproc não é propício para ambientes distribuídos de larga escala. O modelo 1Ptask (ver Seção 5) é mais indicado para ambientes computacionais de larga escala, tipicamente sistemas heterogêneos e compartilhados com servidores *multicores*, comuns nas *grades* e *nuvens* computacionais, pois permite uma maior flexibilidade de execução as tarefas para fazer o balanceamento de carga da aplicação. A versão

autônoma indica colocar o algoritmo da aplicação no modelo 1Ptask, realizar um processo de autoconfiguração da aplicação e incluir a utilização do *middleware* EasyGrid AMS para o gerenciamento autônomo das tarefas. Como ocorre com a maioria das aplicação projetadas com o EasyGrid AMS, os benefícios dessa abordagem incluem adotar políticas (de escalonamento, comunicação, tolerância a falhas) adaptadas às necessidades específicas de cada aplicação, gerando assim um desempenho ainda melhor.

Uma outra questão relacionada ao modelo de programação é a divisão do domínio da aplicação em blocos de tamanhos fixos realizada na versão MPI original. Através da Figura 2, é possível visualizar este problema. A execução sequencial da projeção usando o algoritmo BIOCLIM foi conduzida em uma máquina dedicada, pertencente ao *cluster* (descrito em Seção 7), e, a cada contagem de blocos (1 linha do mapa), o tempo de execução da projeção dele foi exibido. Pelo gráfico da figura, pode-se verificar que os tempos variam bastante, com média de 0,28 segundos e desvio padrão de 0,15, indicando que a quantidade de trabalho de cada bloco é diferente. Por esta razão, a escolha de um tamanho fixo de bloco prejudica o balanceamento de carga visto que o cálculo, para alguns blocos, pode ser rápido e, para outros, demorado.

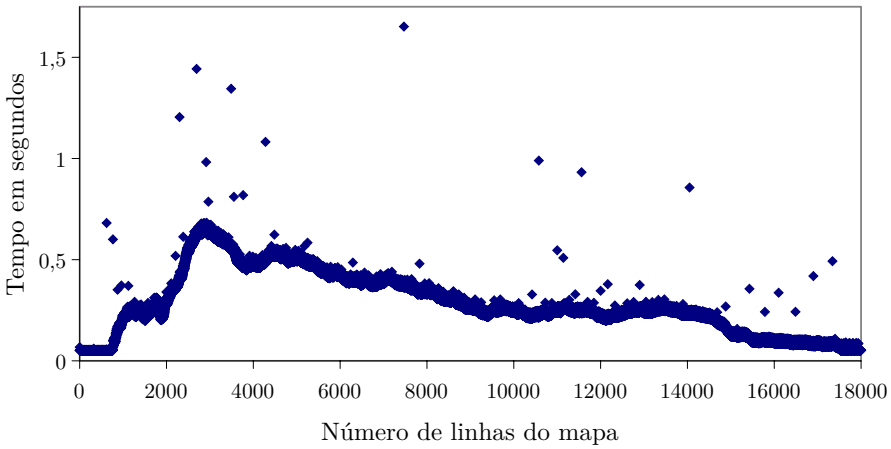


Figura 2. Tempo de execução de cada bloco fixo do domínio da projeção ENM.

A versão EasyGrid AMS usa tamanhos de blocos variáveis, já que a granularidade das tarefas é desconhecida. Este é um processo de reconfiguração dinâmica. Cada tarefa (lembrando que no modelo 1Ptask cada tarefa é representada por um processo) recebe um bloco inicial fixo com o valor de, por exemplo, o número total de pontos no mapa dividido pelo número de processadores. A partir de então, a tarefa calcula a projeção do bloco da mesma maneira que no algoritmo sequencial, mas limita a execução deste cálculo por um temporizador $timeout$. Caso o tempo de execução da projeção do bloco ultrapasse $timeout$, uma ou mais novas tarefas serão criadas para que continuam a projeção deste bloco. O particionamento do mapa é feito dinamicamente e, a cada nível de criação (tarefas iniciais estão no nível 0, seus filhos estão no nível 1 e assim sucessivamente) o valor do temporizador $timeout$ é alterado para um valor inferior. Considerando que $timeout$ representa a granularidade de tempo da tarefa, a razão para a alteração do $timeout$ é tornar a granularidade das tarefas que vão ficando para o final mais fina. Deste modo, as tarefas criadas que vão para a fila de escalonamento do EasyGrid AMS serão distribuídas entre os processadores mais eficientemente [Nascimento et al. 2007a].

O Algoritmo 3 corresponde aos passos de execução de cada tarefa da aplicação de projeção do OpenModeller, chamada de *partitioner*. Como entrada, ele recebe o modelo produzido na fase de modelagem, os dados ambientais, a dimensão Y do mapa original, o bloco e o valor inicial do temporizador $timeout$. Cada tarefa terá uma saída representada por $mapa_{l_i}$, onde l_i é a linha inicial do bloco e é única (identifica a posição do bloco no mapa). As repetições encontradas nas linhas 1 e 2, onde a segunda é aninhada à primeira, representam o percorrimento de cada ponto no bloco de dados do mapa. O valor x varia de l_i a l_j, passados como argumento do algoritmo, que são as linhas inicial e final que delimitam o bloco. O valor y varia de 0 a Dim_y. Assim, os dados ambientais são capturados na posição (x, y) e colocados na variável amb (linha 3). O modelo é, então, aplicado a amb e o valor da probabilidade é colocado em P, na linha 4. Em seguida (linha 5), o valor P convertido para valor de imagem é escrito em $mapa_{l_i}$ na posição $(x - l_i, y)$. Como é uma imagem nova, ela deve começar na coordenada $(0, 0)$ e a expressão para a linha do mapa $x - l_i$ fornece esta configuração.

O trecho do Algoritmo 3 compreendido entre as linhas 1 e 5 apresenta o mesmo comportamento existente no trecho entre as linhas 9 e 13 do Algoritmo 2. Este trecho corresponde ao processamento do problema de projeção ENM. A diferença está na forma como é feito o particionamento (autoconfiguração na versão EasyGrid AMS) e no gerenciamento das tarefas.

Algoritmo 3: Algoritmo das tarefas *partitioner* de projeção com o EasyGrid AMS.

Entrada: $modelo$ - modelo previamente gerado por um algoritmo.
$dados\_ambientais$ - dados ambientais.
Dim_y - dimensão Y do mapa completo.
(l_i, l_j) - bloco a ser calculado.
$timeout$ - temporizador inicial.
Saída: $mapa_{l_i}$ - mapas parciais.

```
1  para x ← l_i a l_j faça
2      para cada y ← 0 a Dim_y − 1 faça
3          amb ← dados_ambientais na posição (x, y)
4          Aplica modelo a amb e coloca o valor em p
5          Escreve p em mapa_{l_i} na posição (x − l_i, y)
6      se temporizar por timeout então
7          Calcula ntarefas
8          Calcula novo timeout
9          Divide bloco restante (x + 1, l_j) por ntarefas e coloca cada sub-bloco em
           S
10         para cada k ∈ S faça
11             Cria tarefa com argumentos modelo, dados_ambientais, Dim_y, k e
               timeout
```

Na linha 6 do Algoritmo 3, após a repetição em relação ao eixo Y (isto é, uma linha completa), é verificado se o temporizador esgotou pelo valor de $timeout$. Caso negativo, o cálculo segue para a próxima linha. Em caso positivo, na linha 6, o número de novas tarefas é gerado de acordo com os dados que foram processados neste tempo $timeout$.

Por exemplo, num total de 100 blocos, se 10 blocos foram calculados em 5 segundos por 1 tarefa, para calcular os restantes 90 blocos é preciso de 9 tarefas ($ntarefas = 9$). Neste caso, o valor de $ntarefas$ é estimado. Na linha 8, o valor de $timeout$, que será passado para as novas tarefas, é decrementado de acordo com o nível de criação das tarefas (tarefas iniciais estão no nível 0, seus filhos estão no nível 1 e assim sucessivamente). A razão para isto é tornar a granularidade das tarefas que vão ficando para o final mais fina e permitir, então, um melhor balanceamento de carga feito pelo EasyGrid AMS. Na linha 9, o restante do bloco é dividido entre as $ntarefas$ tarefas e cada sub-bloco é adicionado ao conjunto S de $ntarefas$ elementos. Por fim, para cada $k \in S$ (linha 10), uma nova tarefa é criada com os argumentos $modelo$, $dados\_ambientais$, Dim_y, k e $timeout$ (linha 11).

No fim de todo esse processo de mapeamento, toda a imagem estará calculada mas dividida entre diversos arquivos de saída. Uma tarefa do tipo *collector* é, então, usada para agrupar os dados em cada arquivo e gerar uma imagem final contendo todo o mapa de projeção.

7. Avaliação Experimental

Dois tipos de experimentos foram realizados: 1) para comparar a versão MPI original com a versão autônoma com EasyGrid AMS e 2) para avaliar a escalabilidade da projeção do openModeller com o EasyGrid AMS. No primeiro tipo de experimento, foi utilizado um *cluster* com 2 máquinas Intel Xeon X5650 com 2 processadores de 6 núcleos, cada, totalizando 24 CPUs, e com 24 GB de memória principal. O segundo tipo de experimento usa um *cluster* que possui 16 máquinas de 8 núcleos cada, totalizando 128 CPUs. Todos estes ambientes de execução estavam dedicados aos experimentos e as entradas usadas para cada algoritmo executado foram as mesmas.

Para os valores apresentados nesta seção de experimentos, a eficiência é calculado considerando a equação: $eficiencia = \frac{speedup}{P} \times 100\%$, onde P é o número de processadores ou CPU e $speedup = \frac{\textit{tempo sequencial}}{\textit{tempo paralelo}}$ é o *speed-up* da execução paralela em relação à versão sequencial disponível na ferramenta OM. A métrica eficiência indica o quanto um algoritmo paralelo utiliza bem os recursos disponibilizados.

7.1. Experimento 1

No experimento 1, P processos trabalhadores são usados para ambas as implementações – original com MPI e autônoma com EasyGrid AMS – em P CPUs. Portanto, na verdade, $P + 2$ processos são usados para a versão original (existem os 2 processos extras). Os algoritmos BIOCLIM, ENVSCORE, GARP, DG_GARP, DG_GARP_BS, MAXENT, GARP_BS, SVM, RF e NICHE_MOSAIC foram escolhidos para serem usados neste experimento, de acordo com seus tempos de execução (os mais altos foram preferidos, exceto o ENVDIST, por durar cerca de 20 dias) e a disponibilidade de suas entradas (por exemplo, não havia a nossa disposição as entradas para o algoritmo AQUAMAPS).

A Tabela 1 apresenta os resultados comparativos de *speed-ups*, em relação ao algoritmo sequencial, obtidos para a versão MPI original e a versão EasyGrid AMS com o uso de vários algoritmos diferentes. Variando o número de processadores (de 4 a 24), a versão autônoma com o EasyGrid AMS apresenta valores significativamente melhores de *speed-ups* e, para a maioria dos algoritmos de modelagem, o valor obtido foi próximo do

Tabela 1. *Speed-ups* obtidos usando as versões MPI original e a autônoma com o EasyGrid AMS.

Algoritmo	Versão	*Speed-up*				
		4	8	12	16	24
BIOCLIM	MPI Original	2,58	4,29	5,54	5,45	5,70
	EasyGrid AMS	3,67	7,15	9,96	13,03	19,21
ENVSCORE	MPI Original	2,51	3,86	5,82	6,82	6,49
	EasyGrid AMS	3,32	6,48	9,35	12,67	18,28
GARP	MPI Original	3,12	4,90	6,85	8,25	8,03
	EasyGrid AMS	3,85	7,57	11,21	14,59	21,58
DG_GARP	MPI Original	2,90	4,74	6,17	7,59	7,49
	EasyGrid AMS	3,67	7,11	10,01	13,10	19,73
DG_GARP_BS	MPI Original	3,11	5,80	7,26	9,68	10,49
	EasyGrid AMS	3,78	7,65	10,87	14,32	21,48
MAXENT	MPI Original	3,46	6,07	7,84	10,38	11,45
	EasyGrid AMS	3,88	7,62	10,99	14,61	21,70
GARP_BS	MPI Original	2,98	5,42	7,50	9,46	10,33
	EasyGrid AMS	3,85	7,60	10,96	14,52	21,77
SVM	MPI Original	3,32	5,63	8,01	10,54	12,59
	EasyGrid AMS	3,90	7,71	11,17	14,85	22,24
RF	MPI Original	3,28	4,99	7,97	10,15	11,30
	EasyGrid AMS	3,59	7,14	10,35	13,78	20,58
NICHE_MOSAIC	MPI Original	3,28	5,68	7,70	7,49	4,63
	EasyGrid AMS	3,94	7,81	10,10	11,48	12,74

número de processadores usados. Para 24 processadores, isto resulta em um valor médio de eficiência maior que 83%, e com eficiências de 93% e 91%, no caso do algoritmo SVM e GARP_BS, respectivamente.

O gráfico da Figura 3 provê uma melhor visualização da comparação dos resultados produzidos por ambas as versões paralelas. Cada grupo de barras com rótulos de 4, 8, 12, 16 e 24 processos em paralelo representam o ganho de desempenho e da eficiência da versão EasyGrid AMS em relação a versão original MPI, isto é, o quanto o *speed-up* da versão em questão foi superior ao *speed-up* da versão original MPI. Cada barra em cada grupo é um algoritmo usado na projeção do openModeller. A versão paralela da projeção com o EasyGrid AMS sempre obtém um *speed-up* maior em relação à versão original MPI e, na maior parte das vezes, é próximo a P. Para a maioria dos algoritmos, conforme o número de processos aumenta, o ganho do EasyGrid AMS em relação ao MPI aumenta também. Para BIOCLIM, com $P = 24$, o *speed-up* da versão da projeção com o EasyGrid AMS chega a ser 3,4 vezes maior que o valor do *speed-up* da versão original MPI.

A escalabilidade é um ponto fraco da versão MPI. Para alguns algoritmos, os *speed-ups* crescem levemente até os 24 processos e, para outros algoritmos (como DG_GARP e NICHE_MOSAIC), o valor diminui com 24 processos comparado a 16 processos. Para uma boa escalabilidade, *speed-up* linear é esperado conforme o número P aumenta. Para a versão EasyGrid AMS, esta propriedade é melhor alcançada embora não seja perfeito. No caso do algoritmo NICHE_MOSAIC, a escalabilidade é mais fraca (o

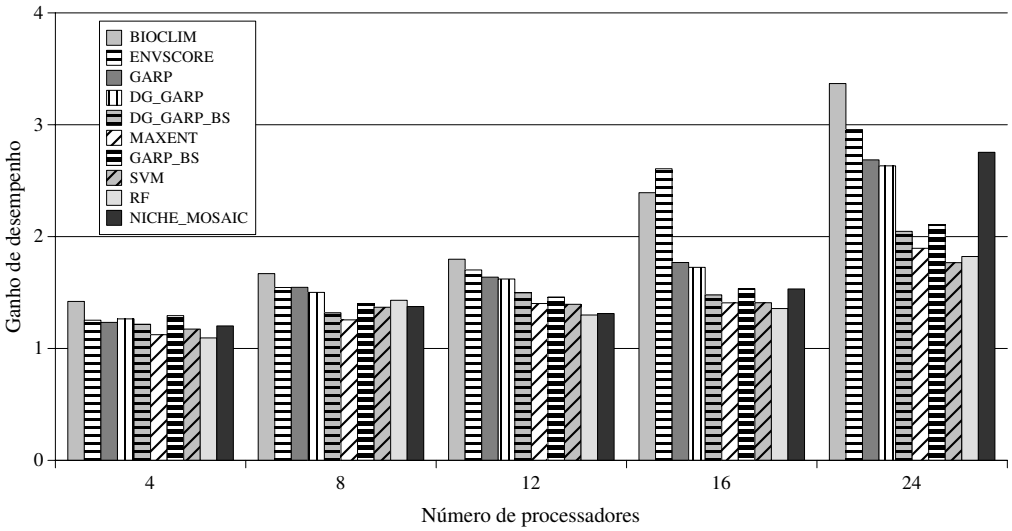


Figura 3. Comparação dos ganhos de desempenho e eficiência da versão EasyGrid AMS em relação a versão paralelo original em MPI.

speed-up é aproximadamente 13 enquanto poderia ser próximo de 24, como acontece para os outros algoritmos) mesmo tendo o *speed-up* quase três vezes melhor que a versão original MPI. Alguma característica do algoritmo não permite uma escalabilidade melhor. Mesmo assim, a eficiência, em termos de utilização dos recursos, é claramente melhor para o EasyGrid AMS uma vez que tem a vantagem devido a sua capacidade de atingir níveis de desempenho mais aceitáveis.

7.2. Experimento 2

No experimento 1, já foi possível concluir que a versão MPI original apresenta um resultado de desempenho e escalabilidade ruins, com até 24 processadores, e projeta-se a queda de desempenho para uma quantidade maior de processadores. Isto se deve ao gargalo de gerenciamento de tarefas que é menos frequente na versão proposta neste artigo. Neste segundo experimento, a escalabilidade da versão autônoma com o EasyGrid AMS é avaliada. Os algoritmos utilizados são ENVDIST e SVM, sendo que ENVDIST é o mais demorado de todos, chegando a quase 20 dias de execução sequencial. O SVM foi escolhido por apresentar o melhor *speed-up* para $P = 24$. Os números de CPUs P utilizados são 24, 48, 96 e 128.

Primeiramente, para se ter uma ideia do número de tarefas criadas durante a execução da projeção ENM, a quantidade total de tarefas para a instância SVM é de aproximadamente 5400, para todos os números de CPUs utilizados, apresentando baixa variância. Para a instância ENVDIST, o quantidade total de tarefas é de aproximadamente 16000, para todos os números de CPUs utilizados, apresentando também baixa variância. O número de tarefas não aumenta conforme o aumento de CPUs porque o objetivo do algoritmo é manter a maior quantidade de tarefas possível, independente do número de CPUs, de modo que elas tenham granularidade fina (mas que sobreponha a sobrecarga de gerenciamento do *middleware*).

A Tabela 2 mostra os tempos em segundos de execução da parte *partitioner*, a mais custosa, e a parte *collector* (concatenação das imagens geradas paralelamente) para o a

projeção usando os algoritmos ENVDIST e SVM. Os tempos de *collector* são próximos entre si para cada algoritmo e parecem não ter relação com o aumento do número P de CPUs, os valores crescem ligeiramente. O tempo total de execução é a soma dos tempos do *partitioner* e do *collector*. Como a implementação do *collector* não é paralelo, o tempo de execução dele pode representar uma parte significante do tempo total de execução. Isto ocorre no caso do algoritmo SVM, onde o tempo de execução total é de 285 segundos e a parte referente ao *collector* corresponde a 16% deste valor.

Tabela 2. Tempos em segundos (*partitioner* e *collector*) obtidos para ENVDIST e SVM.

	ENVDIST		SVM	
P	*partitioner*	*collector*	*partitioner*	*collector*
24	70758,88	75,93	1186,28	41,18
48	35316,90	85,26	614,53	59,50
96	17758,27	88,44	313,29	47,33
128	13390,62	98,59	238,36	46,59

O gráfico representado na Figura 4 mostra a eficiência em relação à execução sequencial de toda a projeção paralela com o EasyGrid AMS para cada um dos dois algoritmos. Nestes resultados, inclui-se o tempo de execução da tarefa *collector* que é executada sequencialmente. Para cada P (eixo horizontal indica o número de processadores), existem duas barras, uma para o ENVDIST e outra para o SVM. A barra cinza escura representa a eficiência de execução para o algoritmo ENVDIST e a barra cinza clara é referente ao algoritmo SVM. Para o algoritmo ENVDIST, a eficiência é alta tanto contando com a parte de *collector* quanto sem ela (ver também a Tabela 2). Neste caso, o *collector* pouco prejudica o desempenho. Ao contrário, para o algoritmo SVM, a eficiência é ligeiramente prejudicada pela tarefa não paralela *collector*, havendo uma diferença representativa entre a eficiência com e sem ela. Uma melhoria seria paralelizar o algoritmo do *collector* também, usando uma estratégia de redução hierárquica, por exemplo. Mesmo assim, as execuções para ambos os algoritmos mostraram-se bastante eficientes.

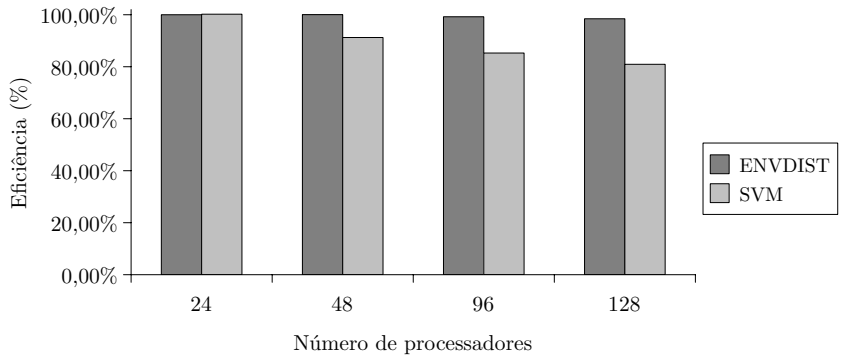


Figura 4. Eficiência da projeção para os algoritmos ENVDIST e SVM usando 24, 48, 96 e 128 CPUs.

8. Conclusão

Neste artigo foi proposto um novo algoritmo paralelo e autônomo para um problema de projeção em domínio geoespacial: a projeção do modelo de nicho ecológico. Este tipo

de aplicação apresenta as características de granularidade desconhecida e independência entre as tarefas, já que as operações sobre o domínio geoespacial são independentes.

A abordagem usada, pelo algoritmo paralelo original da ferramenta openModeller, é do tipo mestre-trabalhador e realiza um particionamento em blocos fixos possibilitando o balanceamento do trabalho entre tarefas no modelo 1Pproc. Já a nova abordagem do algoritmo paralelo e autônomo coloca a aplicação no modelo 1Ptask através de um particionamento de tarefas em blocos de tamanho variável autoconfigurado e utiliza o *middleware* EasyGrid AMS para fazer o gerenciamento das tarefas provendo autootimização e autorrecuperação à aplicação. Além disso, o algoritmo original centraliza os resultados no mestre, o que gera uma grande sobrecarga, já que os resultados são referentes a pedaços de mapas e apresentam uma grande quantidade de dados. No caso do algoritmo proposto, cada tarefa gera seu próprio sub-mapa em arquivo e não apresenta este gargalo durante a execução. A junção dos sub-mapas é feita ao término da projeção. Esta mesma abordagem autônoma pode ser utilizada em outros problemas como, por exemplo, um problema da descoberta de estruturas de proteínas da área de bioquímica [Oliveira and Rebello 2010].

Os resultados mostraram que o algoritmo autônomo com o EasyGrid AMS apresentou melhor desempenho em todos os experimentos comparado ao algoritmo original. A versão original MPI apresenta baixa escalabilidade enquanto a execução do algoritmo proposto é melhor escalável na grande maioria dos casos. Com 24 processadores, os *speed-ups* obtidos com a versão EasyGrid AMS chegam a ser de 177% a 337% maior do que os *speed-ups* da versão original MPI. Além disso, a versão paralela com o *middleware* escala bem nos testes com até 128 processadores, obtendo valores de eficiências perto de 100%, em uma média de 89% entre todos os resultados. Para melhorar os resultados ainda mais, a junção dos sub-mapas deve ser realizada em paralelo futuramente.

Referências

Abrahart, R. and See, L. (2014). *GeoComputation, Second Edition*. CRC Press.

Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., and Saltz, J. (2013). Hadoop GIS: a high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, 6(11):1009–1020.

Amaral, R., Badia, R. M., Blanquer, I., Braga-Neto, R., Candela, L., Castelli, D., Flann, C., De Giovanni, R., Gray, W. A., Jones, A., et al. (2015). Supporting biodiversity studies with the EUBrazilOpenBio hybrid data infrastructure. *Concurrency and Computation: Practice and Experience*, 27(2):376–394.

Boeres, C. and Rebello, V. E. F. (2004). EasyGrid: Towards a framework for the automatic grid enabling of legacy MPI applications. *Concurrency and Computation: Practice and Experience*, 16(5):425–432.

Bonham-Carter, G. F. (2014). *Geographic information systems for geoscientists: modelling with GIS*, volume 13. Elsevier.

EUBrazil Cloud Connect (2015). EUBrazil Cloud Connect. `http://www.eubrazilcloudconnect.eu`.

Geller, G. N. and Melton, F. (2008). Looking forward: Applying an ecological model web to assess impacts of climate change. *Biodiversity*, 9(3-4):79–83.

Leidenberger, S., De Giovanni, R., Kulawik, R., Williams, A. R., and Bourlat, S. J. (2015). Mapping present and future potential distribution patterns for a meso-grazer guild in the baltic sea. *Journal of biogeography*, 42(2):241–254.

Muñoz, M. E. S., De Giovanni, R., Siqueira, M. F., Sutton, T., Brewer, P., Pereira, R. S., Canhos, D. A. L., and Canhos, V. P. (2011). openModeller: a generic approach to species' potential distribution modelling. *GeoInformatica*, 15(1):111–135.

Nascimento, A., Sena, A., Boeres, C., and Rebello, V. E. F. (2007a). Distributed and dynamic self-scheduling of parallel MPI grid applications. *Concurrency and Computation: Practice and Experience*, 19(14):1955–1974.

Nascimento, A., Sena, A., da Silva, J., Vianna, D. Q. C., Boeres, C., and Rebello, V. E. F. (2007b). On the advantages of an alternative MPI execution model for grids. In *CCGRID '07*, pages 575–582, Rio de Janeiro, Brazil. IEEE Computer Society.

Oliveira, F. G. and Rebello, V. E. F. (2010). Algoritmos branch-and-prune autônomos. In *SBRC'10*, pages 335–348, Gramado, Rio Grande do Sul, Brazil.

Oliveira, F. G. and Rebello, V. E. F. (2011). Aplicações Autônomas + Sistemas Simples = Futuro Feliz? In *WoSida, SBRC'11*, Campo Grande, Mato Grosso do Sul, Brazil.

OpenModeller Team (2015). Openmodeller webpage. `http://openmodeller.sourceforge.net/`.

Peterson, A. T., Soberón, J., Pearson, R. G., Anderson, R. P., Martínez-Meyer, E., Nakamura, M., and Araújo, M. B. (2011). *Ecological niches and geographic distributions (MPB-49)*. Princeton University Press.

QGIS Project (2015). QGIS - A Free and Open Source Geographic Information System. `http://www.qgis.org/`.

Ramachandra, T., Kumar, U., Aithal, B. H., Diwakar, P., and Joshi, N. (2010). Landslide susceptible locations in western ghats: Prediction through OpenModeller. In *Proc. of the 26th Annual In-House Symposium on Space Science and Technology, ISRO-IISc*, pages 65–74, Bangalore, India.

Ribeiro, F., Sena, A., Nascimento, A., Boeres, C., and Rebello, V. E. F. (2010). A self-configuring N-body application. In *CIS, SBAC-PAD'10*, Petrópolis, Rio de Janeiro, Brazil.

Silva, J. and Rebello, V. E. F. (2007). Low Cost Self-healing in MPI Applications. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface, 14th European PVM/MPI User's Group Meeting*, pages 144–152, Paris, France. Springer.

Sutton, T., De Giovanni, R., and Siqueira, M. F. (2007). Introducing open modeller – a fundamental niche modelling framework. *OSGeo Journal*, 1(1).

Zarco-González, M. M., Monroy-Vilchis, O., and Alaníz, J. (2013). Spatial model of livestock predation by jaguar and puma in Mexico: Conservation planning. *Biological Conservation*, 159:80 – 87.

Zhao, Y., Padmanabhan, A., and Wang, S. (2013). A parallel computing approach to viewshed analysis of large terrain data using graphics processing units. *International Journal of Geographical Information Science*, 27(2):363–384.

Trilha Principal do SBRC 2016
Sessão Técnica 6
Redes Veiculares

Um Modelo Híbrido para Entrega de Conteúdo em Redes Veiculares

Fabrício A. Silva[1,4], Thais Regina M. B. Silva[1], Eduardo Cerqueira[2], Linnyer B. Ruiz[3] e Antonio A. F. Loureiro[4]

[1]Instituto de Ciências Exatas e Tecnológicas (UFV-Florestal)
[2]Fac. Engenharia da Computação e Telecomunicações (UFPA)
[3]Departamento de Informática (UEM)
[4]Departamento de Ciência da Computação (UFMG)

Resumo. *As aplicações para redes veiculares estão surgindo com o objetivo de tornar o trânsito mais seguro, menos congestionado, mais informativo e prazeroso. Para isso, um dos principais requisitos de tais aplicações é a entrega eficiente de conteúdo aos veículos. Normalmente, duas abordagens têm sido usadas para a entrega de conteúdo em redes tradicionais: Redes de Entrega de Conteúdo (Content Delivery Networks ou simplesmente CDN) e Redes Par-a-Par (P2P). Porém, as características específicas das redes veiculares sugerem que essas abordagens isoladas, como originalmente propostas para a Internet, não são adequadas para esse tipo de rede. Neste trabalho, é proposto um modelo híbrido em que conceitos de CDN e P2P são estendidos e adaptados para as redes veiculares. São definidos e descritos os principais componentes a serem implementados para o modelo proposto, que servem de base para projetistas de aplicações e serviços. Para concluir, é implementada uma solução para demonstrar a eficiência do modelo proposto em relação ao desempenho das aplicações.*

Abstract. *Vehicular network applications are emerging to reality with the objective of making traffic safer, less congested, more informative, and enjoyable. To this end, a fundamental requirement for such applications is the efficient delivery of content. Usually, two approaches have been used to deliver content in the traditional Internet: Content Delivery Networks (CDN) and Peer-to-Peer (P2P). However, several characteristics of VANETs and their applications suggest that pure CDN and P2P models, as originally conceived for the Internet, are not suitable for them. In this work, we propose a hybrid model where concepts from both CDN and P2P are inherited and adapted to vehicular networks. We define and describe the main components to be implemented in the proposed model. Finally, we propose and evaluate a solution to demonstrate that the proposed model is useful for the performance of vehicular applications.*

1. Introdução

As redes veiculares são formadas pela comunicação entre veículos automotores, e entre veículos e unidades de acostamento (RSUs) [Campista et al. 2009]. As aplicações para esse tipo de rede têm evoluído de simples trocas de mensagens de alerta para sistemas avançados com demanda por entrega de variados conteúdos [Costa-Montenegro et al. 2012]. Essas aplicações emergentes requerem que conteúdos digitais heterogêneos sejam entregues aos veículos, incluindo notificações de trânsito, previsão de tempo, vídeos e imagens

de propagandas, vídeos e imagens de entretenimento. No entanto, a tarefa de entrega de conteúdo em redes veiculares não é trivial, e vários desafios ainda devem ser abordados para que as aplicações sejam eficientes nos seus objetivos [Gerla et al. 2014].

Duas abordagens para a entrega de conteúdo são comumente usadas na Internet [Passarella 2012]: Rede de Entrega de Conteúdo (*Content Delivery Networks* ou simplesmente CDN) e Redes Par-a-Par (P2P). As CDNs fazem uso da replicação do conteúdo em servidores estrategicamente localizados, redirecionando as requisições para o servidor de réplica mais apropriado. Com isso, provêem uma alta disponibilidade de conteúdo em uma arquitetura infraestruturada, com a exigência da existência dos servidores, e que os mesmos estejam preparados para receberem as réplicas. Diferentemente, nós em uma rede P2P cooperam entre si, oferecendo os recursos disponíveis para seus pares, levando assim a soluções escaláveis e tolerantes a falhas. Além disso, uma rede P2P geralmente é auto-organizada, necessitando assim de um baixo custo de implantação. No entanto, a descoberta e a entrega de conteúdo pode levar a atrasos significativos e gerar sobrecarga na rede, principalmente em redes com topologia muito dinâmica.

Algumas características das redes veiculares sugerem que os modelos isolados de CDN e P2P, como originalmente concebidos para a Internet, não são apropriados para esse tipo de rede. Primeiramente, muitas aplicações das redes veiculares, conhecidas como *push-based*, não necessitam de requisições explícitas dos clientes, que devem receber um conteúdo potencialmente de seu interesse, como por exemplo uma propaganda em uma determinada região. Além disso, a mobilidade dos veículos faz com que a topologia da rede seja extremamente dinâmica, dificultando assim a busca por conteúdo e o estabelecimento de enlaces fim-a-fim para a sua entrega. Em relação aos servidores estáticos de réplica, sua disposição em cenários urbanos de larga escala é uma tarefa cara em termos de dinheiro e tempo. Por último, alguns conteúdos são dependentes de localização e de tempo, sendo válidos somente dentro de uma área de interesse (AoI) em um período de tempo.

Neste artigo, é proposto um novo modelo híbrido, chamado de Rede Veicular de Entrega de Conteúdo (do inglês *Vehicular Content Delivery Network* ou VCDN), que integra, estende e adapta conceitos de CDN e P2P para as redes veiculares. De um lado, a infraestrutura estática composta por servidores de réplicas é usada para aumentar a disponibilidade de conteúdo. Neste caso, o conceito tradicional de replicação de CDN é estendido para que réplicas sejam alocadas também em veículos, e não somente nos servidores estáticos. Por outro lado, veículos cooperam entre si de maneira similar às redes P2P para a busca e entrega de conteúdo. Essa cooperação também é estendida para estar ciente da mobilidade veicular, um fator fundamental para a qualidade de entrega de conteúdo nas redes veiculares.

Em resumo, o modelo proposto é escalável, tolerante a falhas, ciente de mobilidade, e aplicável tanto em cenários com e sem infraestrutura. Além disso, vale a pena destacar que o modelo é compatível com as recentes arquiteturas orientadas a conteúdo [Silva et al. 2015b, Amadeo et al. 2012]. O modelo VCDN foi implementado em uma solução de entrega de conteúdo em uma área de interesse, e comparado a duas outras soluções da literatura. Os resultados mostram que foi possível aumentar a disponibilidade de conteúdo e ainda assim, reduzir o consumo de recursos da rede.

O restante deste trabalho está organizado da seguinte maneira. A Seção 2 apresenta os principais trabalhos relacionados. Em seguida, a Seção 3 apresenta os detalhes do modelo VCDN proposto. A descrição das avaliações e os resultados estão presentes na Seção 4. Finalmente, as conclusões e alguns trabalhos futuros são descritos na Seção 5.

2. Trabalhos Relacionados

A entrega de conteúdo em redes veiculares é um tema que vem recebendo a atração de pesquisadores nos últimos anos [Silva et al. 2014]. Em linhas gerais, os trabalhos existentes focam em duas áreas principais: replicação de conteúdo e disseminação de conteúdo. Abaixo, os principais trabalhos da literatura para cada uma dessas áreas são descritos, começando pelos relacionados à replicação.

Algumas soluções adotam a estratégia de cálculo de um índice comparável, o que permite aos veículos decidirem localmente quais são mais apropriados para replicar algum conteúdo [Borsetti et al. 2011, Fiore et al. 2013]. Essas propostas implementam algoritmos distribuídos para o cálculo do índice, que geralmente representa as características de mobilidade dos veículos, tais como direção, distância para a região de interesse, velocidade e trajetória. Esse índice fornece uma boa sugestão local de veículos que potencialmente serão boas réplicas. Porém, o índice considera somente uma visão restrita da rede, e o algoritmo distribuído adotado pode gerar uma sobrecarga na rede.

Dois trabalhos da literatura desenvolvem soluções para escolher um veículo para replicar um conteúdo e entregá-lo de um ponto de origem fixo para outro de destino também fixo [Acer et al. 2011, Khabbaz et al. 2012]. Apesar de eficientes para o que foram propostos, os alvos do conteúdo nessas soluções são sempre fixos, o que restringe a sua implantação em VANETs, onde os alvos geralmente são veículos se movendo constantemente. Outras soluções exploram o conhecimento atual e esperado da topologia da rede para fazerem a replicação de conteúdo [Trullols-Cruces et al. 2012, Malandrino et al. 2014]. Apesar de levarem a uma taxa de entrega do conteúdo alta, a necessidade de conhecimento da topologia atual e futura da rede e a adoção de algoritmos complexos de grafos, tornam essas soluções menos atrativas, principalmente em cenários de larga escala. Por fim, outras duas soluções visam a escolha de réplicas de forma centralizada utilizando os pontos de origem e destino dos veículos [Silva et al. 2016, Silva et al. 2015a].

Em relação à disseminação de conteúdo, o principal foco da literatura é no encaminhamento de pacotes com o intuito de não sobrecarregar a rede em uma transmissão broadcast [Meneguette et al. 2014]. A solução [Cao et al. 2014] implementa uma estratégia em que os provedores anunciam para seus vizinhos os conteúdos disponíveis. Assim, um interessado envia uma requisição direta para o provedor, por meio de estações infraestruturadas. Por outro lado, em [Sanguesa et al. 2015], um esquema de disseminação decide a frequência de disseminação com base na densidade e topologia da rede.

Todas as soluções, tanto de replicação quanto de disseminação, possuem vantagens e desvantagens, podendo ser mais adequadas para contextos específicos. Até o momento, não foi proposta nenhuma solução que atenda às demandas de todas as aplicações de redes veiculares, principalmente pelo amplo domínio que engloba essas aplicações. Além disso, também não foi encontrado nenhum modelo ou arcabouço para a camada de aplicação que visa a entrega de conteúdo em redes veiculares.

Neste trabalho, o objetivo é propor um modelo a ser seguido pelos projetistas de aplicações com demanda de entrega de conteúdo. O modelo poderá adotar diferentes estratégias de replicação e disseminação (entrega), sendo flexível para ser implementado de maneira a atender as demandas das aplicações. Com esse modelo, as soluções se tornarão mais flexíveis, expansíveis, e aplicáveis em diferentes cenários.

3. Rede Veicular de Entrega de Conteúdo

Nesta seção, é apresentado o novo modelo híbrido, chamado VCDN, que adapta e estende conceitos de CDN e P2P. Primeiramente, são descritas as características das abordagens puramente CDN e puramente P2P para a entrega de conteúdo em redes veiculares, com o objetivo de apresentar seus pontos positivos e negativos, e motivar a proposta de um modelo híbrido. Em seguida, é proposto o novo modelo na Seção 3.3.

3.1. Abordagem Puramente CDN

Na abordagem puramente CDN, unidades de acostamento (*Roadside Units* ou RSUs) são responsáveis por atuarem como servidores replicadores. Para isso, é necessário uma primeira fase que envolve o planejamento e instalação das RSUs na área de atuação da rede. Em seguida, são escolhidas as RSUs mais adequadas para replicarem um determinado conteúdo. A partir de então, os veículos recebem o conteúdo das RSUs utilizando a comunicação veículo-para-infraestrutura (V2I).

A principal vantagem dessa abordagem para o contexto das redes veiculares é a utilização de recursos da rede infraestruturada para disponibilizar, buscar e entregar conteúdos. No entanto, existe um custo alto para instalar e manter as RSUs de forma a cobrir toda a área da rede, especialmente em se tratando de cenários de larga escala. Além disso, essa solução não é tolerante a falhas de RSUs, já que as mesmas podem levar a áreas descobertas. Por último, a comunicação estritamente V2I é, em geral, mais cara que a veículo-para-veículo (V2V), o que pode levar a um custo maior para os provedores e para os consumidores de conteúdos.

3.2. Abordagem Puramente P2P

A natureza auto-organizável das redes P2P é um atrativo para as redes veiculares, dada a sua topologia extremamente dinâmica. Neste caso, os veículos cooperam entre si formando uma rede *ad hoc* e decentralizada. Conteúdos nesta abordagem puramente P2P são compartilhados entre os veículos, que se unem para disponibilizá-los para seus pares. Para incentivar a cooperação, são dados benefícios para veículos cooperativos, como por exemplo prioridade na entrega de conteúdos.

A principal vantagem de tal abordagem é a tolerância a falhas, pois a falha em algum provedor de conteúdo é suprida por outros na região. Além disso, o custo de instalação é muito baixo, sendo necessário somente que os pares estejam executando um programa em comum. No entanto, o custo pela busca por conteúdo pode ser significativo, em termos de atraso e sobrecarga na rede, dadas as características dinâmicas das redes veiculares, em que provedores de conteúdo se movem e levam o conteúdo para outras regiões. Para dificultar ainda mais, um cliente pode ter se movido quando o provedor é encontrado, fazendo com que a entrega do conteúdo seja ainda mais custosa.

3.3. Modelo Híbrido Proposto

O modelo híbrido proposto (veja Figura 1) explora as vantagens de ambas as abordagens, puramente CDN e P2P, e estende e adapta seus conceitos considerando as características das redes veiculares. Por um lado, a replicação de conteúdo em servidores estáticos (i.e., RSUs) é herdada de CDN para aumentar a disponibilidade de conteúdo. O modelo proposto vai além e estende essa funcionalidade, permitindo a replicação de conteúdo também em veículos com o objetivo de aumentar ainda mais a disponibilidade de conteúdo e tornar o modelo tolerante a falhas (Veja veículo mais escuro na Figura 1). Por outro lado, os conceitos de P2P para busca e entrega distribuída de conteúdo são adotados, além do incentivo a veículos cooperativos. Além disso, para reduzir a sobrecarga na rede e o tempo de busca por um conteúdo, o modelo proposto explora as unidades de acostamento (RSUs) como *trackers* de conteúdos na vizinhança. Por fim, um serviço de gerenciamento de mobilidade é utilizado para monitorar padrões de mobilidade e dar suporte às decisões de replicação, descoberta e entrega de conteúdo.

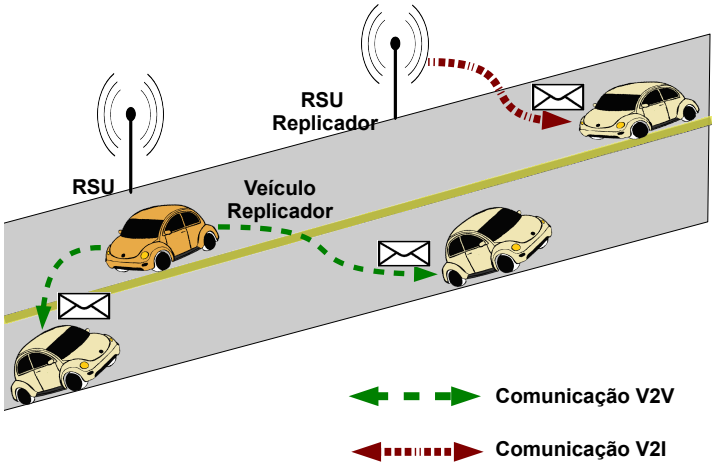


Figura 1. Arquitetura geral do modelo VCDN. Conteúdos são replicados tanto em RSUs quanto em veículos. Um RSU replicador entrega o conteúdo utilizando V2I, enquanto veículos replicadores o fazem por meio da comunicação V2V. Vale ressaltar que, apesar de não ilustrado na figura, o modelo também é compatível com cenários sem RSUs.

O modelo híbrido proposto é tolerante a falhas pois falhas em servidores replicadores, tanto RSUs quanto veículos, são compensadas por outros na região. Além disso, considerando uma replicação efetiva, é esperado que os conteúdos estejam disponíveis perto dos clientes, reduzindo a sobrecarga e o atraso para a busca e entrega dos mesmos. A possibilidade de se ter veículos como replicadores também torna a solução escalável, um fator fundamental para redes veiculares. O modelo também funciona em cenários com e sem infraestrutura, já que a utilização de RSUs não é obrigatória. Por fim, o serviço de gerenciamento de mobilidade leva a tomadas de decisões mais efetivas, pois considera como a mobilidade pode influenciar na entrega de conteúdo.

3.4. Detalhes do Modelo

A Figura 2 ilustra os componentes do modelo proposto, que são flexíveis o suficiente para serem implementados de acordo com as demandas de cada aplicação. Essa flexibilidade é fundamental para as redes veiculares, pois várias aplicações ainda estão surgindo, e aplicações ainda nem pensadas poderão fazer parte desse tipo de rede. Além disso, é importante destacar que cada módulo pode ser executado em diferentes entidades físicas, sendo possível assim a utilização do modelo em diferentes arquiteturas. A seguir, cada um dos componentes é explicado.

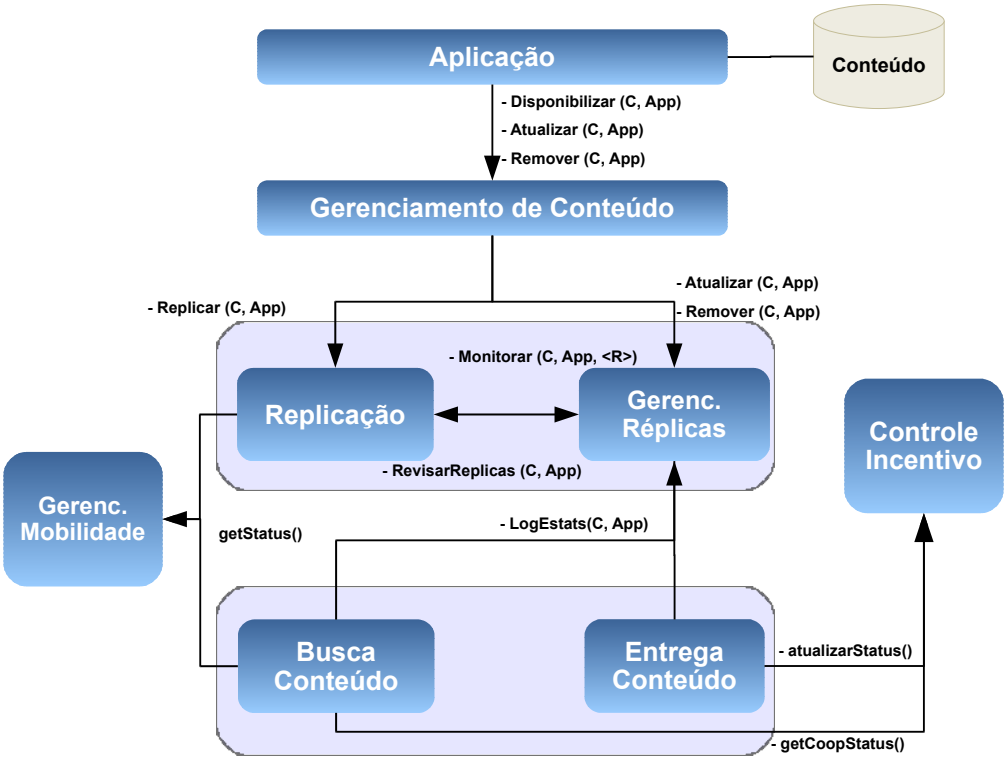


Figura 2. Componentes que devem ser implementados no VCDN. O objetivo é disponibilizar um conteúdo *C* para alguma aplicação *App*. Para isso, replicadores *<R>* são escolhidos para entregar o conteúdo aos interessados.

Gerenciamento de Mobilidade: Em se tratando de redes veiculares, a mobilidade é um fator determinante para as tomadas de decisões. Para ajudar nas decisões, esse módulo é responsável por prover a situação da mobilidade em relação a áreas ou veículos específicos. A situação de mobilidade pode variar de solução para solução, representando por exemplo as trajetórias dos veículos envolvidos, o grafo de contatos atual e esperado da rede, a densidade da rede em determinada região, e os pontos de origem e destino dos veículos. Para aumentar as informações disponíveis nesse módulo, ele pode ser integrado a outros sistemas de monitoramento de trânsito, como por exemplo o proposto em [Júnior et al. 2013].

Gerenciamento de Conteúdo: Este módulo provê serviços para que a aplicação disponibilize, atualize e remova conteúdos. Ao iniciar sua execução, a aplicação provê informações importantes sobre os conteúdos a serem distribuídos, para que o gerenciamento de conteúdo possa então acionar o módulo para replicar os mesmos. Quando

novos conteúdos devem ser disponibilizados, ou conteúdos existentes devem ser alterados ou removidos, esse módulo é acionado. Vale ressaltar pelas interfaces dos módulos que o modelo proposto é compatível com domínios com múltiplas aplicações.

Replicação de Conteúdo e Gerenciamento de Réplica O objetivo da replicação é tornar os conteúdos disponíveis para os potenciais clientes. A decisão de quando e em quais RSUs e veículos replicar depende fundamentalmente das características do conteúdo. Além disso, o padrão de mobilidade veicular é um fator importante, pois os contatos entre veículos entre si, e entre veículos e RSUs, irão determinar a efetividade da entrega aos clientes. Veículos e RSUs selecionados atuam como servidores replicadores, participando ativamente da tarefa de entrega de conteúdo.

O módulo de *Replicação de Conteúdo* é responsável por selecionar veículos e RSUs para atuarem como servidores replicadores. Essa tarefa é fundamental para que os conteúdos sejam disponibilizados próximos aos potenciais clientes, diminuindo assim o tempo e sobrecarga para que seja encontrado e entregue, além de aumentando a satisfação dos clientes. Para isso, a replicação utiliza como fonte de informação o módulo de gerenciamento de mobilidade, para que a escolha dos replicadores seja ciente da mobilidade veicular. Existem algumas soluções que focam especificamente na replicação de conteúdo, sendo que não existe uma solução única que é a melhor para todas as aplicações [Silva et al. 2014]. Mais uma vez, esse componente do modelo híbrido proposto é flexível para adotar a melhor solução para cada caso.

Após a replicação, o módulo de *Gerenciamento de Réplica* inicia o monitoramento dos servidores replicadores. O objetivo é manter a consistência dos conteúdos nas réplicas e acompanhar a efetividade desses servidores. Para isso, os módulos de busca e entrega de conteúdo (descritos abaixo) informam ao gerenciamento de réplica estatísticas sobre a efetividade dos replicadores atuais, como o número de clientes atendidos, a quantidade de dados entregues, e a vazão média aferida, por exemplo. Com isso, quando um servidor replicador não está sendo efetivo, esse módulo invoca o módulo de replicação com o objetivo de alterar a lista de replicadores para melhorar a disponibilidade de conteúdo.

Busca e Entrega de Conteúdo: Nesta etapa, o conteúdo é entregue aos clientes que fizeram requisição explícita (i.e., aplicações *pull-based*) ou não (i.e., aplicações *push-based*). A entrega do conteúdo aos clientes depende fundamentalmente se a aplicação é *push-* ou *pull-based*. Para o primeiro caso, os servidores replicadores devem entregar o conteúdo para todos os veículos alvo, que são aqueles potencialmente interessados. Um desafio nesse caso refere-se a identificar os veículos alvo para algum conteúdo, que pode variar de aplicação para aplicação, como por exemplo algum conteúdo direcionado a veículos em uma área de interesse específica, ou que se dirigem a determinada região.

Por outro lado, clientes em uma aplicação *pull-based* devem requisitar o conteúdo que desejam explicitamente. Neste caso, o módulo de *Busca de Conteúdo* tem um papel fundamental para encontrar os provedores existentes para uma requisição. Para isso, os clientes enviam uma mensagem de *look up* que é disseminada até uma distância em número de passos; essa distância é incrementada até um valor máximo configurável, até que um provedor seja encontrado. Vale destacar que esse módulo é compatível com outras estratégias, que podem ser mais adequadas dependendo da aplicação.

Servidores replicadores, ao receberem uma mensagem de *look up* de um con-

teúdo disponível, respondem com uma mensagem de disponibilidade, que contém, além da informação dos conteúdos disponíveis, suas informações de mobilidade e recursos disponíveis. Esses últimos são informados de acordo com o histórico cooperativo do cliente. Dentre os potenciais provedores, o cliente então escolhe aquele(s) que melhor poderá(ão) atendê-lo, e envia a requisição para o(s) mesmo(s).

Para reduzir o tempo para se encontrar um conteúdo, as RSUs atuam como *trackers* e escutam os *beacons* enviados pelos provedores informando sobre os conteúdos disponíveis. Mecanismos de *beacons* são comuns em redes veiculares [Zeadally et al. 2012], fazendo com que essa funcionalidade seja facilmente habilitada. Com isso, uma RSU é capaz de responder a uma mensagem de *look up*, informando ao cliente sobre um potencial servidor replicador na região.

O módulo de *Entrega de Conteúdo* é responsável por entregar o conteúdo aos clientes. Em uma aplicação *pull-based*, os provedores respondem a mensagens de requisição enviando o conteúdo diretamente ou utilizando comunicação em múltiplos passos. Nesse último caso, outros veículos são explorados como encaminhadores, e são beneficiados por essa cooperação em futuras demandas. Por outro lado, os provedores em aplicações *push-based* enviam o conteúdo periodicamente para alcançar os potenciais clientes com determinadas propriedades.

Vale destacar que esses módulos são compatíveis com modelos recentes orientados a conteúdo [Silva et al. 2015b, Amadeo et al. 2012], em que o conteúdo é representado por um nome e é utilizado o roteamento baseado em nomes.

Controle de Incentivo: Como nas redes P2P, as aplicações que adotam o modelo VCDN requerem a cooperação dos veículos para operar apropriadamente. Por exemplo, veículos devem oferecer seus recursos para atuarem como servidores replicadores em troca de algum benefício. Além disso, veículos devem atuar como encaminhadores de pacotes em transmissões em múltiplos passos. Portanto, o módulo de *Controle de Incentivo* é importante para definir e controlar como veículos cooperativos irão se beneficiar. Para isso, o módulo de *Entrega de Conteúdo* informa a quantidade de dados entregues e encaminhados por cada veículo, que indicam o quão cooperativo cada veículo tem sido. Essa informação é consultada pelo módulo de *Busca de Conteúdo*, em que os provedores podem oferecer mais recursos para veículos cooperativos. Esse módulo é compatível com diferentes estratégias de benefícios, como as baseadas em crédito ou em reputação.

4. Avaliação

Para aferir o desempenho do modelo proposto, foi implementada no simulador de rede OMNET++[1] uma solução seguindo os componentes descritos neste artigo. Abaixo seguem os detalhes das simulações.

Aplicação: A aplicação *push-based* considerada visa disponibilizar uma imagem de propaganda de um estabelecimento comercial em uma área de interesse (AoI) circular com raio r de 500 m na cidade de Ottawa, Canadá. O conteúdo (i.e., imagem) é estático e de tamanho 100 KBytes, devendo ser entregue a todos os veículos que trafegam pela AoI durante o tempo de vida da aplicação. É assumido que toda a área é coberta por RSUs.

[1] http://www.omnetpp.org

Gerenciamento de Mobilidade: As informações de mobilidade utilizadas são os pontos de origem (O) e destino (D) dos veículos. Cada veículo é responsável por fornecer seus pontos de O-D no momento que iniciam seu trajeto. Os outros módulos, com destaque para o de replicação, utilizam essas informações nas tomadas de decisão.

Replicação de Conteúdo: A replicação do conteúdo é o principal fator para manter o mesmo disponível para os veículos alvo. Neste trabalho, a replicação é baseada na proposta descrita em [Silva et al. 2016].

O módulo de replicação de conteúdo executa em um servidor instalado na infraestrutura, interligado a todas as RSUs. Todo veículo, ao partir em direção ao seu destino, envia uma mensagem de *Enter* contendo seus pontos de O-D para a RSU mais próxima, que então a encaminha para o módulo de replicação. Ao receber os pontos de origem o_i e destino d_i de um veículo v_i, esse módulo deve decidir se v_i deve atuar como servidor replicador ou não. Essa decisão é feita com base em duas informações: a distância esperada d_i^A que v_i irá percorrer dentro da AoI, e um índice de cobertura I_i^A durante o período que v_i estará dentro da AoI.

A distância d_i^A é calculada considerando o segmento da reta $\overrightarrow{o_i d_i}$ que intercepta a AoI. Caso $\overrightarrow{o_i d_i}$ não intercepte a AoI, v_i não é considerado como um candidato a replicador. Quanto maior a distância em relação ao raio r da AoI, maior o peso dado para v_i ser considerado como replicado, como definido pela fórmula abaixo:

$$w_i^d = \frac{d_i^A}{2 \times r}. \tag{1}$$

Para o cálculo do índice de cobertura I_i^A, seja $\overline{A_v} = \pi \times r_v^2$ a área de cobertura de cada veículo, dada pelo alcance de transmissão r_v do veículo. Além disso, seja $\overline{A} = \pi \times r^2$ a área total da AoI, dada pelo seu raio r. Assim, é calculada uma estimativa do número n_i^A de réplicas que estarão na AoI simultaneamente a v_i e a área total que essas réplicas são capazes de cobrir no melhor cenário (i.e., com o mínimo de interseção entre os veículos). Como o melhor cenário dificilmente ocorrerá, esperamos a ocorrência de áreas descobertas e outras redundantes, que devem ser diminuídas. Para isso, definimos o índice como a seguinte função negativa quadrática:

$$I_i^A(n_i^A) = (\frac{-1}{10}) \times ((\frac{n_i^A \times \overline{A_v}}{\overline{A}}) \times 100)^2 + 100. \tag{2}$$

O raciocínio é aumentar a chance de um veículo ser escolhido como réplica quando a AoI estiver potencialmente descoberta no mesmo período em que o veículo é esperado de estar dentro da mesma. A função negativa quadrática representa esse raciocínio, já que quanto maior o número de veículos cobrindo a AoI, menor será a contribuição do índice para a escolha de um veículo.

Finalmente, é calculada a probabilidade de um veículo v_i ser escolhido como réplica pela média das duas métricas definidas pelas Equações 1 e 2:

$$p_i = \frac{w_i^d + I_i^A}{2}. \tag{3}$$

Resumidamente, veículos que são esperados percorrer uma distância maior dentro da AoI quando a área estiver com poucas réplicas, são mais prováveis de serem escolhidos como réplicas. Essa solução visa escolher potenciais bons veículos como réplicas e balancear as distribuição das mesmas ao longo to tempo de vida do conteúdo a ser entregue.

Entrega de Conteúdo e Controle de Incentivo: Por se tratar de uma aplicação *push-based*, a entrega é feita por transmissão periódica, a cada 1 s, dos servidores replicadores, com alcance de 1 passo. Vale destacar que cada servidor replicador somente realiza a entrega quando está trafegando dentro da AoI, para evitar que veículos não interessados no conteúdo sejam cobertos, aumentando assim o consumo desnecessário de recursos de rede.

Em relação ao *Controle de Incentivo*, os servidores replicadores são monitorados em relação à quantidade de dados que entregam a outros veículos, e essa informação é armazenada para ser utilizada em seu benefício no futuro. Como essa avaliação considera uma única aplicação, a avaliação do uso do benefício é considerado como trabalho futuro.

Métricas: São medidas duas métricas principais: disponibilidade de conteúdo e custo de comunicação. A primeira avalia o quão disponível o conteúdo está em termos de cobertura (i.e., porcentagem de veículos cobertos). Também foi medida a quantidade de veículos não-alvo (i.e., que não trafegam dentro da AoI), que receberam o conteúdo desnecessariamente. A segunda métrica avalia o custo de comunicação em termos de mensagens trocadas e redundantes, além do número de pacotes perdidos.

Soluções Base: A solução proposta foi comparada a duas outras da literatura: Push-and-Track [Whitbeck et al. 2012] e Linger [Fiore et al. 2013]. Push-and-Track é uma solução centralizada que escolhe veículos disseminadores com base na taxa de cobertura atual. Para isso, cada veículo envia uma mensagem quando entra na rede e mensagens de confirmação (i.e., ACK) de recebimento de conteúdo para o servidor central, que gerencia os veículos alvo que ainda não receberam o conteúdo. Quando esse valor é alto, ele escolhe aleatoriamente outros veículos para serem disseminadores na AoI.

Linger, por outro lado, é uma solução totalmente distribuída em que veículos decidem, por meio da troca de informações locais, quais são mais adequados para atuarem como disseminadores. Cada veículo calcula um índice local com base em sua velocidade e posição atual, e envia para seus vizinhos. Os veículos da vizinhança trocam mensagens e decidem entre eles quais devem disseminar o conteúdo para os potenciais clientes.

O objetivo de escolher essas duas soluções advém da necessidade de comparar nossa solução com outra também infraestruturada, e outra totalmente distribuída. Além disso, essas duas soluções são bem consolidadas na literatura. Diferentemente da solução que segue o modelo VCDN, essas soluções não fazem a replicação e o gerenciamento de réplicas, além de não adotarem informações da mobilidade veicular e mecanismos de incentivo para cooperação.

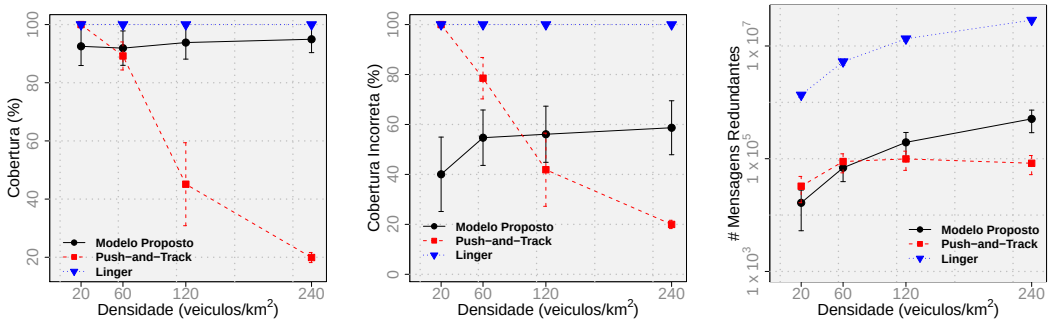
Configurações das Simulação As soluções foram implementadas no simulador de rede OMNET++, que já provê uma implementação do conjunto de protocolos WAVE

(*Wireless Access for Vehicular Environment*) [Morgan 2010] que inclui o padrão IEEE 802.11p para as camadas física e MAC, e o IEEE 1609 para as operações das camadas superiores [Jiang and Delgrossi 2008]. A atenuação de sinal causada por obstáculos e prédios é modelada conforme resultados experimentais reais feitos com o padrão IEEE 802.11p em um cenário urbano, como apresentado em [Sommer et al. 2010].

As taxas de transmissão V2V e V2I são de 1 Mbps e 3 Mbps, respectivamente, e o alcance de transmissão dos veículos é de 100 m com base nos resultados de [Cheng et al. 2007]. Portanto, a área circular de cobertura de um veículo é $\overline{A_v} = \pi \times 100^2$.

Foi adotado um cenário de 9 km^2 do centro da cidade de Ottawa, Canadá, e o tempo de vida da aplicação e do conteúdo é de 3600 s. Para simular uma mobilidade veicular realista, foi utilizado o modelo definido pela ferramenta SUMO[2]. O raio da área de interesse foi fixada em 500 m para ficar de acordo com a área total do cenário. Além disso, as densidades de rede variam de 20 veículos/km^2 a 240 veículos/km^2, para avaliar diferentes cenários. Por fim, todos os resultados representam a média e o intervalo de confiança de 95% dentre 33 rodadas de simulação.

4.1. Resultados



(a) Cobertura (b) Cobertura Incorreta (c) Redundância

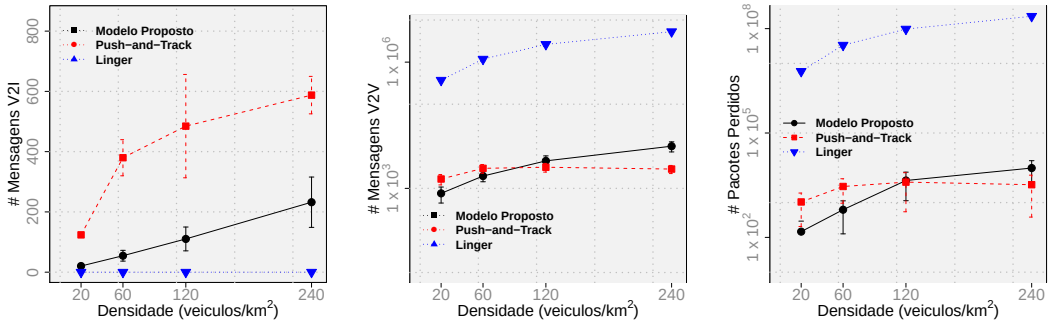
Figura 3. Resultado de cobertura, cobertura incorreta, e redundância.

A Figura 3 apresenta os resultados em termos de cobertura, cobertura incorreta, e redundância. Devido à característica de inundação, Linger obteve 100% de cobertura para todos os cenários, como mostra a Figura 3(a). Porém, essa mesma característica levou Linger a enviar o conteúdo para praticamente todos os outros veículos não-alvo (i.e., que não passam pela AoI), como mostra a Figura 3(b). O modelo VCDN obteve também valores bem próximos de 100% de cobertura, e valores reduzidos de cobertura incorreta (até 60% para redes mais densas). Push-and-Track apresentou resultado satisfatório de cobertura somente para baixas densidades, tendo chegado a apenas 20% de cobertura no cenário mais denso.

O modelo VCDN também se mostrou eficiente em termos de quantidade de mensagens redundantes enviadas, quando comparado às outras soluções, como mostra a Figura 3(c). Isso se deve ao fato da escolha adequada de servidores replicadores, que fazem o balanceamento das entregas ao longo do tempo e espaço. Linger se mostrou pouco efi-

[2]http://sumo-sim.org

ciente para essa métrica, tendo consumido mais recursos de rede (aproximadamente duas ordens de grandeza) para transmitir conteúdo desnecessariamente.



(a) Mensagens V2I (b) Mensagens V2V (c) Pacotes Perdidos

Figura 4. Resultados de mensagens V2I e V2V enviadas, e de pacotes perdidos.

A Figura 4 apresenta os resultados em termos de mensagens de infraestrutura (V2I), mensagens entre veículos (V2V), e pacotes perdidos para cada solução. Push-and-Track precisou de significativamente mais mensagens V2I (sem considerar mensagens ACK) que as outras soluções (aproximadamente 400 mensagens a mais nos cenários mais densos), como mostra a Figura 4(a). Isso se deve ao fato da escolha de disseminadores não ser adequada, precisando se adaptar para aumentar a cobertura quando a taxa de veículos alvo que receberam o conteúdo está baixa. Por ser uma solução totalmente distribuída, Linger não necessita de nenhuma mensagem de infraestrutura. O modelo proposto conseguiu um resultado satisfatório de cerca de 3 vezes menos mensagens V2I que Push-and-Track para todos os cenários. Vale destacar que, em geral, o custo de mensagens V2I é maior que o de mensagens V2V, devido à necessidade da infraestrutura que é controlada por alguma entidade com interesse monetário.

Apesar de não fazer uso de mensagens V2I, Linger necessita de uma quantidade cerca de três ordens de grandeza maior de mensagens V2V do que VCDN e Push-and-Track. Isso leva a uma sobrecarga na rede e, consequentemente, atrasos e perda de pacotes, como mostra a Figura 4(c). O modelo proposto obteve resultados similares a Push-and-Track, tanto na quantidade de mensagens V2V quanto na quantidade de pacotes perdidos.

Em linhas gerais, a implementação do modelo proposto levou a um balanceamento entre a cobertura e o consumo de recursos de rede. Foi possível obter uma taxa de cobertura próxima de 100%, sem precisar de muitas mensagens de infraestrutura (400 mensagens a menos em alguns casos), e sem consumir muitos recursos da rede com mensagens redundantes (duas ordens de grandeza menos). Esses resultados refletem a ideia do modelo VCDN de integrar os principais conceitos de CDN e P2P, para aumentar a disponibilidade de conteúdo, deixando-o mais próximo de seus potenciais clientes. Com isso, os clientes são alcançados mais rapidamente e com um custo menor.

5. Conclusão

Este trabalho apresentou um novo modelo híbrido para a entrega de conteúdo em redes veiculares. O modelo proposto, chamado de VCDN, adapta e estende conceitos de CDN

e P2P, e é flexível para ser utilizado em diferentes aplicações com demandas específicas. Para avaliar a proposta, uma aplicação foi implementada seguindo o modelo, e comparada por meio de simulação com duas outras da literatura. Os resultados mostraram que foi possível balancear a cobertura e o consumo de recursos da rede. Como trabalhos futuros, pretende-se implementar novas aplicações em que requisições devem ser explicitamente enviadas, além de avaliar o mecanismo de controle de incentivo.

6. Agradecimento

Este trabalho contou com o apoio do CNPq (573.738/2008-4 INCT NAMITEC) e CAPES (99999.002061/2014-07).

Referências

Acer, U., Giaccone, P., Hay, D., Neglia, G., and Tarapiah, S. (2011). Timely data delivery in a realistic bus network. In *International Conference on Computer Communications (INFOCOM 2011)*, pages 446–450, Shanghai, China. IEEE.

Amadeo, M., Campolo, C., and Molinaro, A. (2012). CRoWN: Content-Centric Networking in Vehicular Ad Hoc Networks. *IEEE Communications Letters*, 16(9):1380–1383.

Borsetti, D., Fiore, M., Casetti, C., and Chiasserini, C.-F. (2011). An application-level framework for information dissemination and collection in vehicular networks. *Performance Evaluation*, 68(9):876–896.

Campista, M. E. M., Moraes, I. M., Rubinstein, M. G., and Duarte, O. C. M. B. (2009). *Redes Veiculares: Princípios, Aplicações e Desafios*, chapter 5, pages 199–254. Minicursos do XXVII Simpósio Brasileiro de Redes de Computadores.

Cao, Y., Guo, J., and Wu, Y. (2014). Sdn enabled content distribution in vehicular networks. In *International Conference on Innovative Computing Technology (INTECH)*, pages 164–169. IEEE.

Cheng, L., Henty, B., Stancil, D., Bai, F., and Mudalige, P. (2007). Mobile vehicle-to-vehicle narrow-band channel measurement and characterization of the 5.9 ghz dedicated short range communication (dsrc) frequency band. *IEEE Journal on Selected Areas in Communications*, 25(8):1501–1516.

Costa-Montenegro, E., Quiñoy-García, F., González-Castaño, F., and Gil-Castineira, F. (2012). Vehicular entertainment systems: Mobile application enhancement in networked infrastructures. *IEEE Vehicular Technology Magazine*, 7(3):73–79.

Fiore, M., Casetti, C., Chiasserini, C.-f., and Borsetti, D. (2013). Persistent Localized Broadcasting in VANETs. *IEEE Journal on Selected Areas in Communications*, 31(9):480–490.

Gerla, M., Wu, C., Pau, G., and Zhu, X. (2014). Content distribution in VANETs. *Vehicular Communications*, 1(1):3–12.

Jiang, D. and Delgrossi, L. (2008). Ieee 802.11p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040.

Júnior, J. G. R., Quintanilha, I. M., Campista, M. E. M., and Costa, L. (2013). Sistema para monitoramento descentralizado de trânsito baseado em redes veiculares infraestruturadas. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 863–876.

Khabbaz, M. J., Fawaz, W. F., and Assi, C. M. (2012). Which Vehicle To Select? *IEEE Communications Letters*, 16(6):812–815.

Malandrino, F., Casetti, C., Chiasserini, C.-F., and Fiore, M. (2014). Content download in vehicular networks in presence of noisy mobility prediction. *Mobile Computing, IEEE Transactions on*, 13(5):1007–1021.

Meneguette, R. I., Maia, G., Madeira, E. R., Pinto, A. R., Loureiro, A. A., and Villas, L. A. (2014). Um algoritmo autônomo para disseminação de informações em redes veiculares. In *XXXII Simpósio Brasileiro de Redes de Computadores*, pages 911–924.

Morgan, Y. L. (2010). Notes on dsrc & wave standards suite: Its architecture, design, and characteristics. *Communications Surveys & Tutorials, IEEE*, 12(4):504–518.

Passarella, A. (2012). A survey on content-centric technologies for the current internet: Cdn and p2p solutions. *Computer Communications*, 35(1):1–32.

Sanguesa, J. A., Fogue, M., Garrido, P., Martinez, F. J., Cano, J.-C., Calafate, C. T., and Manzoni, P. (2015). Rtad: A real-time adaptive dissemination system for vanets. *Computer Communications*.

Silva, F., Boukerche, A., Silva, T. R. M., Ruiz, L. B., Cerqueira, E., and Loureiro, A. A. (2014). Content replication and delivery in vehicular networks. In *ACM International Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*, DIVANet '14, pages 127–132, New York, NY, USA. ACM.

Silva, F. A., Boukerche, A., Silva, T. R. B., Ruiz, L. B., and Loureiro, A. A. (2016). Geo-localized content availability in vanets. *Ad Hoc Networks*, 36(1):425–434.

Silva, F. A., Boukerche, A., Silva, T. R. M. B., Benevenuto, F., Ruiz, L. B., and Loureiro, A. A. F. (2015a). Odcrep: Origin-destination-based content replication for vehicular networks. *IEEE Transactions on Vehicular Technology*, 64(12):5563–5574.

Silva, V. B. C., Campista, M. E. M., , and Costa, L. H. M. K. (2015b). Uma estratégia de cache proativo para distribuição de conteúdo em redes veiculares. In *XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 403–416.

Sommer, C., Eckhoff, D., German, R., and Dressler, F. (2010). A Computationally Inexpensive Empirical Model of IEEE 802.11p Radio Shadowing in Urban Environments. Technical Report CS-2010-06, University of Erlangen, Dept. of Computer Science.

Trullols-Cruces, O., Fiore, M., and Barcelo-Ordinas, J. (2012). Cooperative Download in Vehicular Environments. *IEEE Transactions on Mobile Computing*, 11(4):663–678.

Whitbeck, J., Lopez, Y., Leguay, J., Conan, V., and de Amorim, M. D. (2012). Push-and-track: Saving infrastructure bandwidth through opportunistic forwarding. *Pervasive and Mobile Computing*, 8(5):682–697.

Zeadally, S., Hunt, R., Chen, Y.-S., Irwin, A., and Hassan, A. (2012). Vehicular ad hoc networks (VANETS): status, results, and challenges. *Telecommunication Systems*, 50(4):217–241.

CO-OP: Uma Solução para a Detecção, Classificação e Minimização de Congestionamentos de Veículos utilizando Roteamento Cooperativo

Allan Mariano de Souza[1], Daniel Guidoni[2], Leonardo C. Botega[3] e Leandro A. Villas[1]

[1]Instituto de Computação – UNICAMP
allan@lrc.ic.unicamp.br, leandro@ic.unicamp.br

[2]Departamento de Ciência da Computação – UFSJ
guidoni@ufsj.edu.br

[3]Centro Universitário Eurípides de Marília – UNIVEM
botega@univem.edu.br

Abstract. *Most large cities suffer with congestion problem, one of the main causes of congestion is the sudden increase of vehicle traffic during peak hours, mainly in areas with bottlenecks. Current solutions in the literature are based on perceiving road traffic conditions and re-routing vehicles to avoid the congested area. However, they do not consider the impact of these changes on near future traffic patterns. Hence, these approaches are unable to provide a long-term solution to the congestion problem, since when suggesting alternative routes, they create new bottlenecks at roads closer to the congested one, thus just transferring the problem from one point to another one. With this issue in mind, we propose an intelligent traffic cooperative routing application called CO-OP, which improves the overall spatial utilization of a road network and also reduces the average vehicle travel costs by avoiding vehicles from getting stuck in traffic. Simulation results show that our proposal is able to forecasting congestion and re-route vehicles properly, performing a load balance of vehicular traffic.*

Resumo. *A maioria das cidades grandes sofrem com problema de congestionamento, uma das principais causas de congestionamento é o aumento do tráfego de veículos durante o horários de pico, principalmente em áreas com gargalos de infraestrutura. Existem soluções atuais na literatura baseadas em detecção nas condições de tráfego e em redirecionamento de veículos para evitar a áreas congestionadas. No entanto, essas soluções não consideram o impacto dessas mudanças nos próximso padrões de tráfego. Assim, essas abordagens são incapazes de fornecer uma solução de longo prazo para o problema do congestionamento, desde quando sugerindo rotas alternativas, criam novos gargalos em um futuro próximo, assim, apenas transferindo o problema de um ponto a outro. Com esta questão em mente, propomos uma solução cooperativa de roteamento chamado CO-OP, o qual melhora a utilização do espaço global e como consequencia reduz o tempo de viagem, evitando congestionamentos. Os resultados das simulações mostram que a nossa proposta é capaz controlar congestionamento e previnir o surgimento de congestionamentos em um futuro próximo.*

1. Introdução

Devido ao crescimento urbano de maneira desorganizada, as grandes cidades estão enfrentando sérios problemas socioeconômicos. Esse crescimento urbano, tipicamente, possui impacto significativo nas estruturas da cidade, uma vez que a demanda por diversos serviços, entre eles os sistemas de transportes, que fazem parte do cotidiano das pessoas [Naphade et al. 2011]. Com relação a esse serviço em particular, estima-se que 23% dos moradores da cidade de São Paulo gastam no mínimo duas horas para chegar ao destino[1]. Existem vários impactos negativos relacionados aos congestionamentos em uma cidade. Além da diminuição da atividade econômica, aspectos como aumento da emissão de CO_2, qualidade do ar entre outros [Bauza and Gozalvez 2013, Karagiannis et al. 2011] também podem ser considerados.

Com o avanço da tecnologia de comunicação sem fio, comunicação móvel e com a padronização das redes veiculares, é possível o desenvolvimento de sistemas de transporte inteligentes (ITS) [Wang et al. 2015]. Um sistema de transporte inteligente integra dados, comunicação e computação para resolver diversos problemas nos sistemas de transportes tradicionais de grandes cidades, visando, por exemplo, reduzir engarrafamentos, aumentar a capacidade das estradas, reduzir acidentes, reduzir custos dos sistemas de transportes, diminuir o tempo de viagens e aumentar a qualidade e segurança dos deslocamentos. Nesse contexto, uma rede veicular possui um papel importante para a realização dos sistemas de transporte inteligentes através de soluções que consideram a comunicação entre carros, *road side units* (RSU), sensores e outros dispositivos com capacidade de comunicação [Maia et al. 2013]. Um sistema de transporte pode criar padrões na movimentação dos carros baseados na coleta de dados on-line, tal como densidade de veículos na rede, velocidade, tempo de viagem entre outros com o objetivo de detectar, controlar e reduzir congestionamentos [Pan et al. 2012, Doolan and Muntean 2013, Bauza and Gozalvez 2013, Wang et al. 2015, Brennad et al. 2015]. O principal desafio no gerenciamento de congestionamento é modificar a rota de alguns veículos sem causar congestionamentos em outros lugares da rede.

Atualmente, existem aplicações comerciais, como o Waze [WAZE], que disponibilizam rotas alternativas baseadas nas informações sobre o trafego disponibilizadas por seus usuários. Entretanto, essas aplicações disponibilizam novos trajetos individuais e dependem das informações coletadas das redes sociais, que podem não representar o real estado das condições de engarrafamento além de não possuírem um pequeno tempo de resposta em situações emergenciais, causadas por um acidente, por exemplo. Para explorar a coleta de informações em tempo real, várias abordagens foram propostas com o objetivo de encontrar as melhores rotas individuais para contornar congestionamentos [Doolan and Muntean 2013, Pan et al. 2012]. Entretanto, ao encontrar novas rotas individuais, novos congestionamentos podem ser criados em diferentes regiões da cidade. Para minimizar esse problema, algumas abordagens projetam *k* melhores rotas, e a seleção da rota escolhida pode ser feita de maneira aleatória [Pan et al. 2012] ou utilizando métodos probabilísticos [Brennad et al. 2015].

Para abordar os problemas mencionados, este trabalho propõe uma abordagem para a prevenção e controle de congestionamentos baseado em roteamento cooperativo.

[1]http://agenciabrasil.ebc.com.br/geral/noticia/2015-09/tempo-gasto-no-transito-ultrapassa-duas-horas-para-23-dos-paulistanos.

Para isso, é descrita uma abordagem de classificação de tráfego em diferentes níveis. Após, o roteamento cooperativo é realizado considerando as condições globais de mobilidade do tráfego de veículos com o objetivo de planejar rotas que não irão gerar novos congestionamentos. Dessa forma, a abordagem proposta nesse trabalho é capaz de planejar em tempo real novas rotas, previnir a criação de novos congestionamentos e melhorar as condições de fluxo de carros nas vias a partir de uma perspectiva global da mobilidade. A abordagem proposta é comparada com três diferentes abordagens da literatura que abordam o problema estudado nesse trabalho, que são *WithRouting* [Brennad et al. 2015], DSP and RkSP [Pan et al. 2012]. Os resultados de simulação mostram que a abordagem proposta é mais eficiente na prevenção de congestionamentos e criação de novas rotas considerando o balanceamento de trafego de veículos nas vias.

Este trabalho está organizado da seguinte maneira. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a nova abordagem para o problema de prevenção e controle de congestionamentos. Os resultados de simulação são apresentados na Seção 4. A Seção 5 apresenta a conclusão e trabalhos futuros.

2. Trabalhos Relacionados

Os autores em [Bauza and Gozalvez 2013] propõem um sistema de cooperação veicular baseado na comunicação entre veículos para detectar congestionamento utilizando lógica Fuzzy, chamado de CoTEC (COperative Traffic congestion detECtion). O sistema proposto utiliza mensagens de alerta, onde todos os veículos enviam periodicamente para seus vizinhos informações sobre as condições de tráfego. Após um veículo receber informações sobre o tráfego, um algoritmo baseado em lógica fuzzy é executado para detectar se existe congestionamento na rodovia onde o veículo está. Quando um congestionamento é detectado por um veículo, o mesmo envia essa informação para seus vizinhos. Após receber várias informações sobre congestionamentos, os veículos, de maneira colaborativa, conseguem caracterizar o congestionamento detectado. Entretanto, ao enviar mensagens periódicas sobre condições do tráfego e a disseminação de congestionamento detectado, o sistema CoTEC pode diminuir a capacidade de fluxo de dados da rede. Apesar de ser capaz de detectar congestionamentos, o CoTEC não considera mecanismos para evitar, diminuir ou controlar o congestionamento. Além disso, o sistema proposto opera somente em rodovias.

Uma solução para sistemas de transporte inteligentes que coleta informações em tempo real, detecta e gerência o congestionamento é proposto em [Brennad et al. 2015]. O sistema proposto utiliza um conjunto de RSU distribuídas na cidade com capacidade de obter uma cobertura de comunicação de todas as ruas. Cada RSU é responsável por gerenciar e detectar congestionamentos de veículos em sua área de cobertura. O sistema proposto utiliza uma mensagem de controle que, periodicamente, realiza a atualização de todas as rotas do veículos para que os mesmos não encontrem congestionamentos. Entretanto, o sistema proposto não gerência o congestionamento quando o mesmo acontece, já que as rotas serão atualizadas na fase de atualização de rotas.

Outra solução para a aquisição em tempo real de informações sobre veículos é proposto em [Pan et al. 2012]. A solução proposta, projetada de maneira centralizada, obtém informações sobre a localização geográfica dos veículos, suas velocidades e direção do percurso para detectar congestionamentos. Quando o veículo se aproxima de um

congestionamento detectado, o sistema calcula uma nova rota baseada em dois algoritmos diferentes, que são: *Dynamic Shortest Path* (DSP), onde as rotas são calculadas combinando o menor caminho e o menor tempo de viagem e *Random k Shortest Paths* (RkSP), onde *k* rotas são calculadas e aleatoriamente o sistema escolhe uma rota para o veículo. O objetivo da escolha aleatória entre *k* rotas é não causar novos engarrafamentos em outros pontos da cidade. O sistema proposto em [Pan et al. 2012] possui a mesma desvantagem do sistema proposto em [Brennad et al. 2015], onde as novas rotas são calculadas apenas na fase de atualização de rotas.

[Doolan and Muntean 2013] propõem o EcoTrec, um algoritmo de roteamento baseados nos conceitos ecológicos que utiliza comunicação V2V. O algoritmo considera as características dos engarrafamentos para melhorar o consumo de combustível e diminuir a emissão de gás carbônico, um dos principais problemas dos engarrafamentos do ponto de vista ecológico. O EcoTrec funciona da seguinte forma. Os veículos disseminam para a rede suas informações de trajeto e consumo de combustível durante sua viajem. De posse desses dados, o sistema cria um grafo, onde as interseções são consideradas vértices e as ruas são consideradas as arestas. Toda a aresta possui um peso, que é calculado a partir das informações disseminadas pelos veículos. Após a criação do grafo, o sistema informa a melhor rota, em termos de consumo de combustível, que deve ser utilizada pelo veículo. Entretanto, o sistema EcoTrec não é capaz de detectar congestionamentos, calculando apenas novas rotas para os veículos. Além disso, o sistema utiliza um algoritmo de caminho mínimo em grafos para o cálculo da nova rota, fazendo com que engarrafamentos possam ser criados em pontos diferentes da cidade, uma vez que vários veículos podem ter a mesma rota com o menor consumo de combustível.

Com base nos problemas das soluções encontradas na literatura, que são: (i) sobrecarga de comunicação [Bauza and Gozalvez 2013]; (ii) visão limitada (local) da movimentação de carros [Pan et al. 2012, Doolan and Muntean 2013, Brennad et al. 2015] e (iii) atraso na criação de novas rotas para evitar congestionamentos [Pan et al. 2012, Doolan and Muntean 2013, Brennad et al. 2015], o presente trabalho propõe uma nova solução cooperativa eficiente para a prevenção e gerenciamento de congestionamentos.

3. CO-OP

Nesta seção é descrito CO-OP (Uma Solução para a Detecção, Classificação e Minimização de Congestionamentos de Veículos utilizando Roteamento Cooperativo), um sistema de transporte inteligente que utiliza uma abordagem cooperativa para roteamento de veículos que estão dentro de uma área de roteamento (*RA - Routing Area*) de alguma via congestionada, definida pelo sistema, para minimizar e controlar congestionamentos em centros urbanos. A RA é uma área utilizada para evitar roteamentos desnecessário de veículos, de modo que apenas sejam roteados os veículos com maior urgência, ou seja, veículos que estão mais próximos do congestionamento e possuem maior probabilidade de ficarem presos no mesmo. O restante da seção é divida da seguinte forma: a arquitetura do CO-OP é descrito na Subseção 3.1. A Subseção 3.2 apresenta o modelo usado para representar o cenário e os veículos. A Subseção 3.3 descreve o método utilizado para classificar a condição de tráfego em cada rua do cenário. Por fim, a Subseção 3.4 apresenta o algoritimo de roteamento cooperativo, o qual faz um balanceamento do tráfego para minimizar e controlar congestionamentos de acordo com a condição de tráfego em cada rua do cenário.

3.1. Arquitetura do Sistema

O CO-OP é divido em quatro partes: (i) Coleta de informações; (ii) Classificação do trágefo; (iii) Análise de rota e; (iv) Sugestão de rota. A Figura 1 descreve a arquitetura do CO-OP e como cada módulo interage entre si.

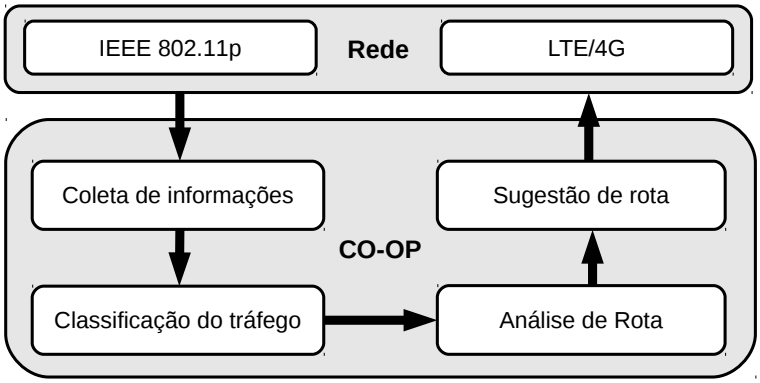


Figura 1. Arquitetura do sistema CO-OP organizada em quatro módulos principais, sendo: Coleta de informações, Classificação do trágefo, Análise de rota e Sugestão de rota.

1. ***Coleta de informações***: módulo reponsável por receber as informações dos veículos (ID, posição atual, rota e velocidade) e manter os dados sobre o tráfego (velocidade média e densidade) atualizados. O módulo *coleta de informações* recebe as informações dos veículos que são enviadas periódicamente atravéz da rede, a qual pode implementar não só o protocolo IEEE 802.11p, mas também LTE/4G;

2. ***Classificação do tráfego***: recebe as informações atualizadas obtidas pelo módulo de *coleta de informações* e periódicamente faz a *classicação do tráfego* de todas as vias do cenário. O intervalo de classificação (*CI - Classification Interval*) é definido pelo sistema;

3. ***Análise de rota***: após a *classificação do tráfego* de todas as vias do cenário, o módulo *análise de rota* analisa quais veículos devem ser re-roteados. Para isso, para cada via congestionada o sistema define uma área de roteamento (RA). Além disso, é verificado se um veículo esta dentro de uma área de roteamento (RA) e se ao menos uma via congestionada esta contida na rota do veículo. Caso essa condição for satisfeita, o veículo precisa ser roteado, caso contrário o veículo permanece com sua rota original;

4. ***Sugestão de rota***: calcula uma nova rota utilizando a abordagem cooperativa proposta para os veículos selecionados no módulo *análise de rota*. A abordagem cooperativa visa melhorar o fluxo do cenário todo, assim não calculando as melhores rotas para cada veículo, mas sim as melhores rotas para manter um tráfego contínuo em todo o cenário. Por fim, a nova rota é enviada para os veículos através da rede utilizando tando protocolo IEEE 802.11p ou LTE/4G.

3.2. Modelo de Representação da Rede

A rede é representada por um grafo ponderado e dirigido $G = (V, E)$, onde o conjunto $V = \{v_1, v_2, \ldots, v_i\}$ representa o conjunto de cruzamentos do cenário (vértices), enquanto o conjunto $E = \{e_1, e_2, \ldots, e_i\}$ corresponde ao conjunto de ruas do cenário (arestas). Além disso, $W = \{w_1, w_2, \ldots, w_i\}$ é o cunjunto de pesos de cada rua $e_i \in E$. Seja $N = \{n_1, n_2, \ldots, n_j\}$ o conjunto de veículos (nós) na rede e $R = \{r_1, r_2, \ldots, r_j\}$ o conjunto de rotas para cada veículo $n_j \in N$.

Todos os veículos n_j na rede periódicamente enviam uma mensagem $mgs^j \leftarrow \{\text{posição}^j, \text{rota}^j, \text{velocidade}^j\}$ para o CO-OP. Assim que novas informações estão disponíveis, CO-OP calcula a velocidade média s_i e a densidade d_i de cada rua $e_i \in E$. Depois disso, atualiza o peso de cada rua, $w_i \leftarrow (s_i, d_i, ms_i)$, onde ms_i é a velocidade máxima da rua e_i. Além disso, baseado nos pesos de cada rua, CO-OP classifica o nível de congestionamento de todo conjunto E periódicamente definido por um intervalo CI e cria um conjunto $C = \{c_1, c_2, \ldots, c_i\}$ onde C é um subconjunto de E. Por fim, para cada rua congestionada c_i é criado uma área de roteamento RA_i e, para todos os veículos n_j dentro de uma RA_i onde pelo menos uma rua e_i de sua rota r_j esteja contida no conjunto C, é calculada uma nova rota para que o veículo evite o congestionamento.

O peso w_i para cada rua e_i é calculado utilizando a Equação 1. A equação foi desenvolvida para o peso ser inversamente proporcional a condição de tráfego de veículos. Dessa forma, quanto melhor a condição do tráfego, menor é seu peso. Portanto, ruas congestionadas possuem maior peso do que rua com fluxo livre de veículos. A Equação 1 é definida da seguinte forma:

$$w_i = 1 - \frac{s_i}{d_i \cdot ms_i} \tag{1}$$

onde s_i, d_i e ms_i são definidas pela rua e_i e, representam respectivamente a velocidade média, desidade e velocidade máxima da rua e_i.

3.3. Classificação dos Níveis de Congestionamento

Detectar e classificar níveis de congestionamentos não é um processo simples. Devido à complexidade de tratar diversas informações, tais como: acidentes, questões relacionadas ao clima, problemas de sinalização, horários de pico e gargalos na infraestrutura de transporte. Neste contexto, há diversas abordagens para detecção de congestionamento [Zhong et al. 2008, Souza et al. 2014, Brennad et al. 2015] e outras abordagens que focam na classicação da condição de trágefo utilizando lógica *fuzzy* [Bauza and Gozalvez 2013, Araujo et al. 2014]. Diferentemente destes trabalhos, CO-OP implementa o algoritmo *k*-NN (*K-Nearest Neighbor*) [Larose 2005] para classificar os níveis de congestionamentos em todo cenário. Além disso, como o CO-OP mantém as informações atualizadas do cenário todo, o mesmo é ciente das condições de tráfego globais, permitindo um roteamento cooperativo dos veículos visando melhorar a condição do tráfego a longo prazo. Isto evita a criação de novos congestionamentos, o que pode acontecer quando o roteamento utilizado calcula as melhores rotas para cada veículo sem balanceamento de tráfego global, como consequência, os congestionamentos se deslocam para outros locais.

Em geral, o *k*-NN precisa de uma base de dados de amostra para treinar o classificador. Para isso, foi empregado um conjunto de dados sintéticos construído de acordo

com a *Highway Capacity Manual* (HCM) [Board 2010]. O HCM contém conceitos, diretrizes e procedimentos para medir a capacidade, o desempenho e a qualidade do tráfego, com base na velocidade e densidade de veículos nas estradas. O conjunto de dados foi construído com base nos Níveis-de-Serviços (*LOS - Level-of-Service*) presentes no HCM. O LOS representa uma medição da qualidade utilizado para descrever as condições operacionais dentro de um fluxo de tráfego. Seis diferentes níveis de serviço são definidos, *LOS A* representa condições de fluxo livre e *LOS F* representa congestionamento intenso. Cada nível define a velocidade mínima e máxima com base na velocidade máxima da estrada. Com o conjunto de dados de treinamento empregado, é possivel identificar quatro condições de tráfego, onde cada um deles é baseado em limites de velocidade apresentados nos LOS da HCM. Além disso, para eliminar falsos positivos, utilizou-se a densidade de veículos nas estrada em combinação com a velocidade média para definir uma condição de tráfego. A densidade é baseada na porcentagem de veículos na estrada de acordo com a sua capacidade máxima. Portanto, a classificação dos níveis de congestionamento é definida da seguinte forma *fluxo livre* = 0, *congestionamento leve* = $1/3$, *congestionamento moderado* = $2/3$ e *congestionamento intenso* = 1. Tabela 1 mostra a classificação dos níveis de congestionamento.

Tabela 1. Classificação da condição de tráfego

Velocidade	Densidade			
	Baixa	Média	Alta	Muito Alta
Rápida	Fluxo livre	Fluxo livre	Fluxo livre	Leve
Média	Fluxo livre	Fluxo livre	Leve	Moderado
Lenta	Fluxo livre	Fluxo livre	Moderado	Moderado
Muito lenta	Leve	Moderado	Moderado	Intenso

A classicação é feita periódicamente em um intervalo pré-definido CI. Portanto, para fazer a classicação, o classificador recebe como entrada a velocidade média s_i e a densidade d_i e disponibiliza como resultado o nível de congestionamento baseado na base de dados, como mostrado na Tabela 1. Sendo assim, após a classicação, CO-OP sabe de todas as ruas congestionadas no cenário.

3.4. Roteamento Cooperativo

O roteamento cooperativo ocorre depois da classificação dos níveis de congestionamento. Assim, novas rotas são calculadas para os veículos dentro de uma área de roteamento que passarão por uma rua congestionada. No entanto, o cálculo das novas rotas é feito em uma perspectiva global para melhorar o fluxo no cenário todo. Diferente do cálculo de melhor caminho, onde a melhor rota é calculada para cada veículo individualmente, o roteamento cooperativo visa evitar o congestionamento e equilibrar o tráfego em vez de descobrir melhores rotas para cada veículo. Assim alguns veículos podem ser penalizados, mas minimiza a probabiliade de ocasionar novos congestionamentos o que é comun nas abordagens de melhor caminho.

A Figura 2 mostra todo o procedimento realizado pelo CO-OP. Primeiramente, os veículos enviam suas informações para o CO-OP, assim que o sistema recebe essas informações, o mesmo atualiza os pesos das arestas no grafo. Além disso, o sistema verifica se deve realizar a classicação (Figura 2 *label* A). Caso não for, o sistema contiua

a receber as informações do veículos, caso contrário o mesmo faz a classificação de todas as ruas do cenário e em seguida define as respectivas áreas de roteamento.

Logo após, o CO-OP verifica quais veículos estão dentro de alguma área de roteamento (Figura 2 *label* B). Para os veículos que não estão dentro de uma área de roteamento, a rota original é mantida para os mesmos. Entretanto, para os veículos que estão dentro de uma área de roteamento, é verificado se os mesmos irão passar por algum congestionamento (Figura 2 *label* C). Se a condição for satisfeita, uma nova rota é calculada.

Durante o cálculo da rota, CO-OP verifica a rota, a posição atual e o destino de todos os veículos. Além disso, para todos os veículos que irão passar por uma rua congestionada e estão dentro de uma área de roteamento, todos os possíveis caminhos a partir da posição atual do veículo até seu destino são calculados. O roteamento cooperativo implementa um algoritmo de busca gulosa. A partir daí, para construir uma nova rota, CO-OP não só verifica todos os caminhos possíveis que o veículo pode percorrer a partir da sua posição atual, com exceção dos caminhos que impedem o veículo de chegar ao seu destino, mas também seleciona a rua com menor peso como próxima rua da nova rota. Além disso, depois de selecionar a próxima rua da nova rota, CO-OP atualiza o peso da rua selecionada e o procedimento seleção continua até que construir toda a rota (Figura 2 *label* D e *label* E).

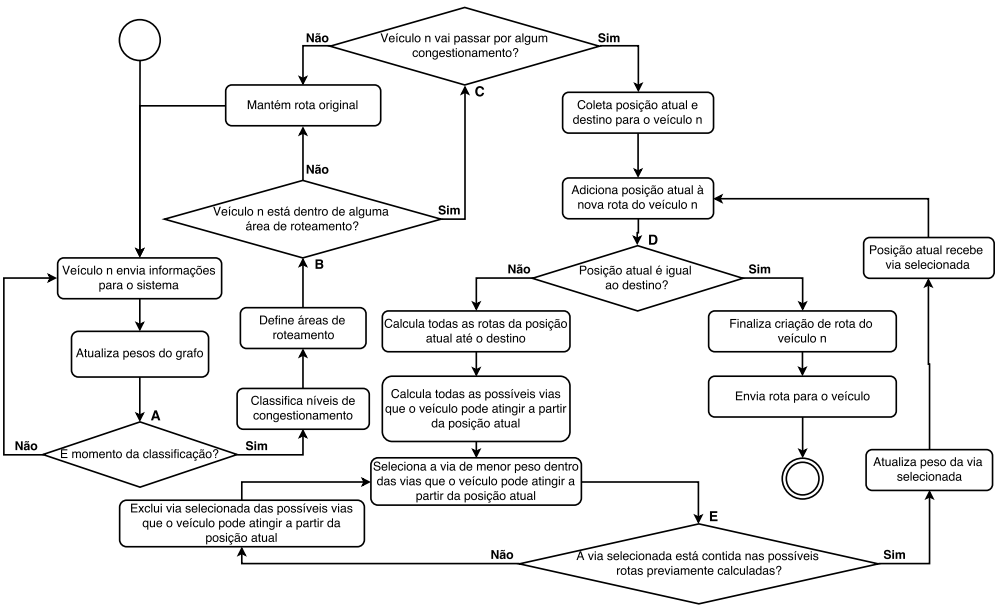


Figura 2. Fluxo de operação do CO-OP para detecção de congestionamentos, roteamento e controle de veículos

A atualização do peso da rua selecionada é feito de acordo com o impacto que aquele veículo terá na mesma. Dessa forma, a atualização do peso é feito utilizando a Equação 2

$$w_i = w_i + (w_i \cdot \frac{l_n^j}{l_e^i}) \tag{2}$$

onde, w_i é o peso atual da rua e_i, l_n^j é o comprimento do veículo n_j e l_e^i é o comprimento

da rua e_i. Selecionar ruas com menor peso melhora a eficiência do tráfego evitando ruas congestionadas e o processo de atualizar o peso da rua selecionada é importante para balancear o tráfego e contribuir para um fluxo de tráfego contínuo.

A Figura 3 descreve como o cálculo de rota é feita pelo CO-OP. A figura mostra o planejamento de rota para um veículo que vai passar por uma estrada congestionada com a posição atual no vértice A e destino no vértice H. Na Figura 3(a) depois de calcular todas as possíveis rotas de A à H, CO-OP inicia o planejamento da rota. A partir do vértice A o veículo pode alcançar B e E (ambos estão dentro de todos os caminhos possíveis calculado anteriormente). No entanto, a aresta que conecta vértice A e E tem o menor peso, portanto, esta aresta é selecionada como a próxima aresta da rota do veículo e seu peso é atualizado. Na Figura 3(b) a partir do vértice E o veículo pode atingir os vértices C, D, F e G, mas os vértices D e F são excluídos dos possíveis próximos vértices porque eles não fazem parte das possíveis rotas calculadas anteriormente. Além disso, entre os vértices C, G e H a aresta que conecta vértice E e o vértice C tem o menor peso, por isso é selecionada como a próxima aresta e seu peso é atualizado. Por fim, na Figura 3(c), a partir do vértice C o veículo pode chegar aos vértices G e H e ambos estão em vias possíveis, no entanto, a aresta que conecta vértice C e H tem o menor peso, portanto, é selecionado como próxima aresta a aresta que conecta vértice C e H e o seu peso é atualizado e, a nova rota calculada é $r_j \leftarrow \{A, E, C, H\}$. Além disso, como o peso das estradas que formam a nova rota foram atualizados, o novo percurso para outro veículo com a mesma posição atual e destino pode ser $r_j \leftarrow \{A, B, E, C, G, H\}$. No entanto, usando um outro algoritmo, sem perspectiva global para construir a nova rota, como *Shortest Path*, a rota para dois ou mais veículos com origem no vértice A e destino no vértice H, as duas novas rotas seriam $r_j \leftarrow \{A, E, H\}$.

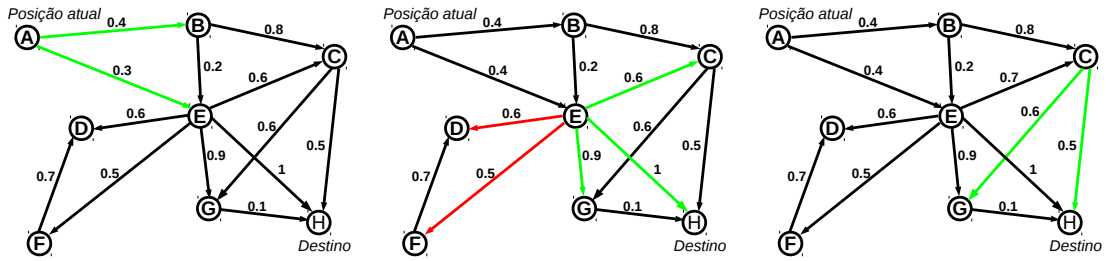


Figura 3. Cálculo de uma rota para um veículo com posição atual no vértice A e com destino no vértice H

4. Resultados

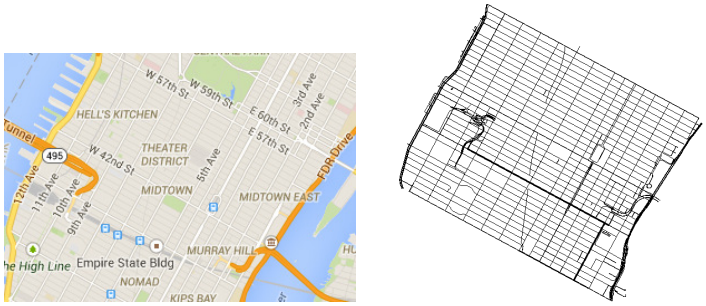
Esta seção descreve a metodologia aplicada, as ferramentas utilizadas, o cenário de simulação e os resultados de simulações comparando o CO-OP com as soluções da literatura DPS, RkSP e With routing.

4.1. Metodologia

Para nossa avaliação, foi utilizado o simulador de mobilidade SUMO (*Simulator for Urban MObility*) [Behrisch et al. 2011], versão 0.17.0. A implementação do gerenciamento do tráfego no SUMO é feita pelo TraCI (*Traffic Control Interface*) [Wegener et al. 2008],

o qual permite o gerenciamento em tempo real durante a simulação. Além disso, ele provê acesso e controle à objetos de simulação, alterando seu comportamento em tempo de execução. Para calcular as emissões de CO_2 e o consumo de combustível, foi utilizado o modelo EMIT implementado pelo SUMO. EMIT é um modelo estatístico baseado na fórmula HBEFA[2], a qual calcula as emissões de CO_2 e o cunsumo de combustível baseado na aceleração e velocidade dos veículos.

É importante salientar que foi utilizado um cenário real para as simulações, retirado do OpenStreetMap[3]. Foi selecionado uma parte da cidade de Manhattan Nova York, Estados Unidos, com um tamanho de 5 km^2, como mostra a Figura 4. A mobilidade dos veículos foi gerada de forma aleatória com as seguintes densidades: 250, 500, 750 e 1000 veiculos/km^2.



(a) Mapa real do OpenStreetMap (b) Mapa no SUMO

Figura 4. Mapa de Manhattam.

A Tabela 2 apresenta os parâmetros de simulação utilizados. Por uma questão de simplicidade, foi utilizado o algoritmo de KNN como classificador com k=5, valor no qual obteve-se melhores resultados em nossas avaliações preliminares. Por fim, para cada análise, os resultados representam a média de 33 replicações com intervalo de confiança de 95%.

Tabela 2. Parâmetros de simulação

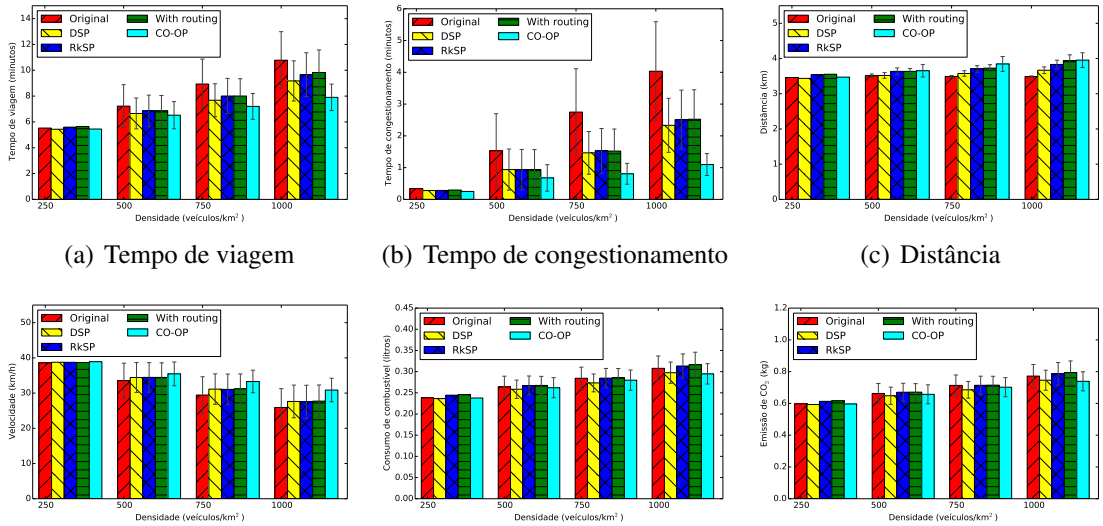
Parâmetros	Valores
Cenário	5 km^2
Intervalo de classificação (CI)	30 s
Área de roteamento (RA)	1000 m
Densidade	250, 500, 750, 1000 v/km^2
# vizinhos mais próximos	5
Intervalo de Confiança	95%

CO-OP foi comparado com o tráfego de veículos original, isto é, abordagem tradicional sem a presença de nenhum mecanismo de re-roteamento de veículos. Além disso, CO-OP foi comparado com três soluções encontradas na literatura, sendo elas: DSP e RkSP [Pan et al. 2012] e With routing solução proposta por Brennand et al. [Brennad et al. 2015]. Para analisar a performance das soluções, as seguintes métricas

[2]http://www.hbefa.net
[3]http://www.openstreetmap.org/

foram utilizadas: *i)* velocidade média: velocidade média dos veículos durante a simulação; *ii)* tempo médio de viagem: tempo médio em que os veículos levaram para atingir seu destino; *iii)* tempo de congestionamento: tempo médio em que os veículos ficaram presos em um congestionamento; *iv)* distância percorrida: distância média percorrida por todos os veículos; *v)* consomo de combustível: consumo médio de combustível de todos os veículos; e *vi)* emissão de CO_2 : emissão média de CO_2 emitida por todos os veículos na simulação.

4.2. Avaliação de Desempenho



(a) Tempo de viagem (b) Tempo de congestionamento (c) Distância

(d) Velocidade média (e) Consumo de combustível (f) Emissão de CO_2

Figura 5. Avaliação de desempenho do CO-OP em comparação com soluções da literatura.

As soluções DSP, RkSP and With routing necessitam de uma configuração inicial de alguns parâmetros relacionados ao intervalo de roteamento. Foram feitos vários testes com diferentes valores de intervalo de roteamento e o melhor intervalo de roteamento encontrado para o cenário avaliado foi de 300 segundos. Além do intervalo de roteamento, a solução RkSP e With routing necessitam encontrar k menores caminhos mínimos para realizar o roteamento. Após vários testes com diferentes valores de k, ambos algoritmos encontraram os melhores resultados para $k = 5$.

A Figura 5 ilustra os resultados de todas as métricas para a avaliação de desempenho considerando diferentes densidades de veículos na rede. Para todos os resultados, pode-se observar que, a medida que a densidade da rede aumenta, o tempo de viagem, distância percorrida, consumo de combustível, emissão de CO_2 e tempo de congestionamento também aumentam e a velocidade média diminui. O objetivo de variar a densidade de carros é "estressar"o cenário avaliado para verificar os resultados dos algoritmos em cenários de congestionamentos. Os resultados descritos abaixo consideram uma densidade de 1000 veículos por km^2 para efeito de comparação entre as soluções.

Como pode ser encontrado, o solução que não modifica as rotas dos veículos (Original) possui uma média de distância percorrida de 3.4 km, uma média de tempo gasto de 10.77 minutos e uma média de velocidade de 25.9 km/h considerando uma densidade

de carros de 1000. Entretanto, usando essa solução, os veículos gastam cerca de 37% do tempo de viagem parados em engarrafamentos (Figura 5(b)). Em comparação com a solução Original, o DSP diminui a média do tempo de viagem em aproximadamente 14%. Essa redução é causada pela periodicidade de troca de rotas dos veículos. A velocidade média é aumentada em aproximadamente 6%. Por outro lado, a modificação das rotas aumenta em aproximadamente 5% a distância percorrida quando o DSP é utilizado (Figura 5(c)). O DSP modifica as rotas dos veículos utilizando rotas ótimas individuais, fazendo com que novos congestionamentos sejam criados. Considerando o DSP, os veículos gastam, aproximadamente, 25% do tempo de viagem em engarrafamentos (Figura 5(b)).

Diferentemente do DSP que seleciona rotas ótimas individuais, o algoritmos RkSP e With routing selecionam *k* rotas, evitando futuros congestionamentos. Entretanto, os resultados mostram que RkSP, em comparação com o Original, diminui o tempo de viagem em 10%, aumenta a distância percorrida em 10% e aumenta a velocidade média em 6%. O aumento da distância percorrida deve-se ao fato da escolha aleatória de uma das *k* rotas existentes, ou seja, possivelmente uma rota com tamanho superior ao tamanho da rota ótima. Mesmo com essa política de seleção de rotas, os veículos, utilizando o RkSP, gastam 26% do tempo de viagem em congestionamentos. Para contornar o problema relacionado a escolha aleatória de rotas, o algoritmo With routing utiliza um método probabilístico na escolha das novas rotas dos veículos. O método probabilístico tem como objetivo o balanceamento do trafego das novas rotas. Em comparação com o Original, a solução With routing diminui o tempo de viagem em 8%, aumenta a distância percorrida em 13% e aumenta a velocidade média em 7%. O tempo gasto em engarrafamento pelo With routing é de 26%.

Diferentemente das outras abordagens, o CO-OP detecta e controla o engarrafamento com o objetivo de criar novas rotas que não gerem novos engarrafamentos em outros pontos da cidade utilizando, ao mesmo tempo, um roteamento colaborativo. O tempo gasto em congestionamentos pelo CO-OP (Figura 5(b)) é cerca de 13% do tempo de viagem, inferior aos 26% do With routing, 26% do RkSP, 25% do DSP e 37% quando nenhuma estratégia é utilizada (Original). Também como consequência da estratégia utilizada pelo CO-OP, a velocidade média dos veículos é de 30 km/h, aproximadamente 20% superior comparado com as demais soluções.Por outro lado, para evitar futuros congestionamentos na criação de novas rotas, a distância percorrida quando a solução CO-OP é utilizada é cerca de 13% superior em comparação quando nenhuma rota é alterada.

Finalmente, os resultados relacionados ao consumo de combustível e emissão de CO_2 são mostrados nas Figuras 5(e) e 5(f). Esses resultados estão diretamente relacionados ao tempo de viagem e o tempo que os carros ficaram parados em congestionamentos. Pode-se perceber que quando nenhuma rota é alterada (Original), o consumo de combustível 0.30 litros para realizar o percurso. O DSP, RkSP, With routing e CO-OP possuem um consumo de 0.29, 0.31, 0.31 e 0.29 litros respectivamente. Em comparação com todas as soluções, o CO-OP é a solução que consome menos combustível para realizar o mesmo percurso. A emissão de CO_2 está diretamente relacionada com o consumo de combustível. A emissão de CO_2 pelas soluções Original, DSP, RkSP e With routing são, respectivamente, 0.77, 0.74, 0.78 e 0.79 superior em comparação com a solução proposta CO-OP.

Ao realizar uma comparação quando a rede possui 250 e 1000 veículos, foi obtido os seguintes resultados. Para a solução que não modifica as rotas (Original), o tempo de viagem, tempo de congestionamento, consumo de combustível e emissão de CO_2 aumentaram, respectivamente, 105%, 911%, 25% e 26%, entretanto, para a velocidade média teve uma redução de 35% e a distância percorrida permanece a mesma. Considerando o DSP, RkSP e With routing, que obtiveram resultados similares, o aumento, considerando a mesma sequência das métricas foi de aproximadamente 70%, 534%, 25% e 26%, porém a velocidade média teve uma redução de 30% e a distância percorrida aumentou aproximadamente 10%. Considerando a solução proposta nesse trabalho (CO-OP), o aumento, considerando a mesma sequência das métricas foi de 27%, 172%, 20% e 21%, entretanto a velocidade média foi reduzida em apenas 19% e a distância aumentou em apenas 13%. Dessa forma, é importante observar que, ao aumentar a densidade de veículos na rede, a solução CO-OP obteve a menor variação das métricas avaliadas, demostrando que o roteamento cooperativo é uma solução interessante na prevenção e gerenciamento de congestionamentos nos cenários avaliados.

5. Conclusões

Neste trabalho, foi proposto CO-OP, uma solução para detcção, classificação e controle de congestionamentos utilizando um roteamento cooperativo. A solução proposta visa reduzir o tempo de viagem, tempo de congestionamento. Os resultados das simulações mostram a eficácia da nossa solução. Quando comparado com mobilidade original dos veículos, o CO-OP reduz o tempo médio de viagem em aproximadamente 37%, e a média do tempo de congestionamento em 72%, aumentando a distância percorrida em apenas 13%. Como trabalho futuro, pretende-se analisar o desempenho da nossa solução em cenários mais realistas, utilizando *traces* de ambientes reais.

6. Agradecimentos

Os autores gostariam de agradecer o apoio financeiro concedido da CAPES e FAEPEX para o bolsista de doutorado Allan Mariano de Souza. Ainda, os autores também agradecem a FAPEMIG e CNPq por financiarem partes dos seus projetos de pesquisas. Por fim, Leandro Villas agradece o apoio financeiro da FAPESP por meio do processo nº 2016/05392-2, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências

Araujo, G., Queiroz, M., Duarte-Figueiredo, F., Tostes, A., and Loureiro, A. (2014). Cartim: A proposal toward identification and minimization of vehicular traffic congestion for vanet. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–6.

Bauza, R. and Gozalvez, J. (2013). Traffic congestion detection in large-scale scenarios using vehicle-to-vehicle communications. *Journal of Network and Computer Applications*, 36(5):1295 – 1307.

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). SUMO - Simulation of Urban MObility: An Overview. In *International Conference on Advances in System Simulation (SIMUL '11)*, pages 63–68.

Board, T. R. (2010). *HCM 2010 - Highway capacity manual*. National Research Council.

Brennad, C., Souza, A., Maia, G., Boukerche, A., Loureiro, A., and Villas, L. (2015). An intelligent transportation system for detection and control of congested roads in urban centers. In *Computers and Communications (ISCC), 2015 IEEE Symposium on*.

Doolan, R. and Muntean, G. (2013). Vanet-enabled eco-friendly road characteristics-aware routing for vehicular traffic. In *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, pages 1–5.

Karagiannis, G., Altintas, O., Ekici, E., Heijenk, G., Jarupan, B., Lin, K., and Weil, T. (2011). Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *Communications Surveys Tutorials, IEEE*, 13(4):584–616.

Larose, D. T. (2005). *k-Nearest Neighbor Algorithm*, pages 90–106. John Wiley and Sons, Inc.

Maia, G., Rezende, C., Villas, L. A., Boukerche, A., Viana, A. C., Aquino, A. L., and Loureiro, A. A. (2013). Traffic aware video dissemination over vehicular ad hoc networks. In *Proceedings of the 16th ACM International Conference on Modeling, Analysis, Simulation of Wireless and Mobile Systems*, MSWiM '13, pages 419–426, New York, NY, USA. ACM.

Naphade, M., Banavar, G., Harrison, C., Paraszczak, J., and Morris, R. (2011). Smarter cities and their innovation challenges. *Computer*, 44(6):32–39.

Pan, J., Khan, M., Sandu Popa, I., Zeitouni, K., and Borcea, C. (2012). Proactive vehicle re-routing strategies for congestion avoidance. In *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, pages 265–272.

Souza, A. M. d., Boukerche, A., Maia, G., Meneguette, R. I., Loureiro, A. A., and Villas, L. A. (2014). Decreasing greenhouse emissions through an intelligent traffic information system based on inter-vehicle communication. In *Proceedings of the 12th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac '14, pages 91–98, New York, NY, USA. ACM.

Wang, M., Shan, H., Lu, R., Zhang, R., Shen, X., and Bai, F. (2015). Real-time path planning based on hybrid-vanet-enhanced transportation system. *Vehicular Technology, IEEE Transactions on*, 64(5):1664–1678.

WAZE. Waze app bugs and issues. `https://wiki.waze.com/wiki/Waze_App_Bugs_and_Issues`.

Wegener, A., Piórkowski, M., Raya, M., Hellbrück, H., Fischer, S., and Hubaux, J.-P. (2008). Traci: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*, pages 155–163. ACM.

Zhong, T., Xu, B., and Wolfson, O. (2008). Disseminating real-time traffic information in vehicular ad-hoc networks. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1056–1061.

RAN: Um Mecanismo para Tratar o Efeito de Sincronização no Padrão WAVE IEEE 802.11p

Erick A. Donato[1], **Edmundo R. M. Madeira**[1], **Leandro A. Villas**[1]

[1]Instituto de Computação – UNICAMP
erick@lrc.ic.unicamp.br, {edmundo,leandro}@ic.unicamp.br

Abstract. *Vehicular networks are seen as the basis for the development of intelligent transportation systems(ITS). Therefore, and in order to take advantage from the benefits of these transportation systems, the development of standards that meet the specific characteristics of vehicular networks were required. The Wireless Access in Vehicular Environments (WAVE) was developed in order to fulfill this need. This standard presents an architecture, based on a division into multiple channels with each channel set for certain types of application and that uses a switching mechanism for the selection of channels, since only one channel is active at a given time. However, in certain scenarios, this channel switching mechanism approach used in WAVE introduces an undesirable effect that allows different vehicles to transmit simultaneously, resulting in collisions. This paper proposes a solution to this problem, with the development of a mechanism based on recalculation of transmission delay, that will work along with other features, such as the broadcast suppression mechanism, in order to ensure greater performance in transmissions.*

Resumo. *As redes veiculares se enquadram como a base principal para o desenvolvimento de sistemas de transporte inteligentes(ITS). Assim sendo, e no sentido de tirar proveito dos benefícios destes sistemas de transporte, tornou-se necessário o desenvolvimento de padrões que levassem em consideração as características específicas das redes veiculares. O Wireless Access in Vehicular Environments (WAVE) foi desenvolvido como forma de proporcionar uma resposta a esta necessidade. Este padrão apresenta uma arquitetura baseada em múltiplos canais, que são definidos de acordo com os tipos de aplicações, e que utiliza um mecanismo de salto de canais para a seleção de canais considerando que em cada período de tempo um único canal estará ativo. No entanto, verificou-se que, em determinados cenários, essa abordagem de salto de canais introduz um efeito de sincronização indesejado, que permite que diferentes veículos transmitam ao mesmo tempo, resultando em colisões. Este trabalho propõe uma solução para este problema, com o desenvolvimento de um mecanismo baseado na atualização do atraso de transmissão que visa trabalhar em parceria com outros mecanismos como o de supressão de broadcast para garantir um maior desempenho nas transmissões.*

1. Introdução

As perspectivas de implementação de ITS's têm despertado de forma significativa o interesse dos pesquisadores, da indústria e dos governos, devido aos benefícios que os mesmos podem proporcionar para a sociedade. O principal objetivo de um ITS é melhorar

a eficiência, a segurança e o prazer em sistemas de transporte, utilizando possivelmente diferentes tecnologias. Uma tecnologia promissora para a implementação de ITS é a rede veicular. Em redes veiculares, os veículos são equipados com sensores, unidades de processamento e interfaces de comunicação sem fio para permitir a comunicação com outros veículos e unidades pertencentes à infraestrutura da rede (*RSU-Road Side Unit*), assim, permitindo a criação de uma rede móvel enquanto os veículos se movem nas estradas e rodovias [Villas 2014].

Uma característica importante das redes veiculares é que suas diferentes aplicações estão estreitamente relacionadas entre si. Por exemplo, reduzir o número de acidentes leva à redução do número de congestionamentos, que por sua vez leva à diminuição do nível de impacto no ambiente. Na tentativa de tirar proveito das vantagens inerentes às redes veiculares, foi atribuído um espaço no espectro eletromagnético [Hartenstein 2008] para a comunicação entre veículos. Além disso, estes estão sendo equipados com rádios de comunicação sem fio que implementam o novo padrão de comunicação *Wireless Access in Vehicular Environments* (WAVE) [IEEE 2010].

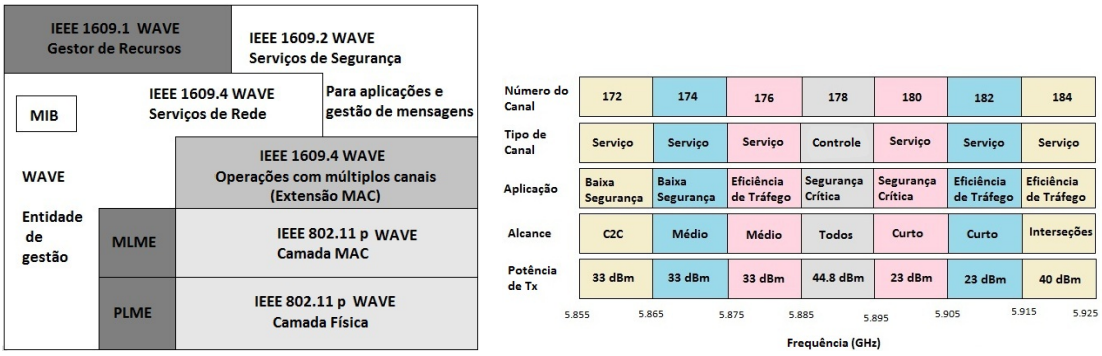
O padrão WAVE apresenta uma arquitetura baseada em múltiplos canais que inclui um canal de controle e seis canais de serviço. Essa arquitetura visa proporcionar o tratamento adequado aos diferentes tipos de aplicações, que vão desde aplicações críticas em termos de requisitos de segurança e tolerância ao atraso, a aplicações menos exigentes nestes contextos, levando sempre em consideração as características próprias das redes veiculares [Campolo 2011, Barradi 2010]. Como o padrão WAVE não obriga a utilização de múltiplas antenas, um mecanismo de salto de canais tem sido utilizado. A multiplexação por divisão de tempo é aplicada nessa abordagem. No entanto, a utilização desse mecanismo de salto de canais introduz um efeito indesejado de ressincronização em protocolos de disseminação de dados *broadcast*. De forma a evitar esse efeito indesejado, este trabalho propõe um mecanismo chamado ***RAN*** (***R**esynchronization **A**ware of the **N**eighbourhood*). Este mecanismo pode ser facilmente acoplado às principais soluções de disseminação de dados encontradas na literatura. Desta forma, tais soluções tornam-se cientes do mecanismo de salto de canais empregado pelo padrão WAVE. Dentre os principais benefícios em se utilizar tal mecanismo pode-se citar a diminuição do número de colisões de pacotes de dados e, consequentemente, um aumento na taxa de entrega dos pacotes.

2. Padrão de Comunicação em Redes Veiculares

Baseado no padrão IEEE 802.11p [IEEE 2010], o *Wireless Access in Vehicular Environments* (WAVE) [IEEE 2010] define uma arquitetura de comunicação especialmente desenhada para as redes veiculares. O WAVE foca nas camadas mais baixas da pilha de protocolos e apresenta uma definição das camadas físicas e de acesso ao meio (MAC) fortemente baseada nos padrões anteriores de redes sem fio [Eichler 2007, Grafling 2010].

A Figura 1(a) mostra em alto nível a arquitetura WAVE. Os componentes são dedicados às funções de rede e de gerenciamento, com destaque para aspectos relacionados com a segurança, o gerenciamento de recursos e o suporte para múltiplos canais de operação. Além disso, também merecem destaque os blocos denotados por MLME (*MAC Layer Management Entity*) e PLME (*Physical Layer Management Entity*) que representam as entidades de gerenciamento das camadas MAC e física. Estes dois blocos

correspondem à separação lógica existente entre as funções de gerenciamento das camadas MAC e física, respectivamente, tendo em conta que essas duas camadas do WAVE foram padronizadas separadamente.



(a) Arquitetura básica do WAVE (b) Subdivisão do WAVE em múltiplos canais

Figura 1. Padrão WAVE [Grafling 2010].

O padrão IEEE 1609.4 define a operação do WAVE com base em múltiplos canais e diferencia os canais em termos das características das aplicações que cada canal suporta. Além da utilização de frequências diferentes, as potências máximas de transmissão também variam de acordo com os canais. A potência máxima de transmissão é atribuída ao canal de controle, reservado às aplicações críticas em termos de segurança, enquanto que para as aplicações menos críticas em termos de segurança são atribuídos os canais com menores níveis de potências de transmissão, os quais são chamados de canais de serviços. A Figura 1(b) mostra a configuração do padrão WAVE subdividida em múltiplos canais e as respectivas configurações.

No padrão IEEE 1609.4, o canal de controle (CCH) e os seis canais de serviço (SCH) são utilizados com base em multiplexação por divisão de tempo. O canal de controle é servido em intervalos de tempo alternados, enquanto que os intervalos restantes de tempo são distribuídos entre os canais de serviço de acordo com as necessidades das aplicações alocadas à cada canal. Devido à baixa tolerância ao atraso que caracteriza algumas aplicações, por exemplo, as aplicações para evitar colisões, o período máximo de multiplexação foi estabelecido em 100 milissegundos.

Adicionalmente ao esquema de funcionamento de múltiplos canais, um esquema de prioridades semelhante ao implementado pelo padrão IEEE 802.11e, conhecido como EDCA, também é implementado na camada MAC do WAVE. Para cada canal, existem quatro categorias de acesso denotadas por AC0, AC1, AC2 e AC3. O menor nível de prioridade corresponde ao AC0 e o maior ao AC3 [Grafling 2010]. Os pacotes de dados, pertencentes a diferentes categorias de acesso, são colocados em filas de espera a partir dos quais são servidos de acordo com um mecanismo de seleção interno. As configurações dos tempos de espera das filas variam segundo a categoria de acesso de cada canal. Durante a fase de seleção de pacotes para transmissão, as quatro categorias de acesso competem internamente e então, o pacote selecionado compete externamente pelo canal com os pacotes de outros veículos que estejam dentro do raio de comunicação.

A camada física do padrão IEEE 802.11p utiliza a multiplexação por divisão ortogonal de Frequência (OFDM) e é similar ao padrão 802.11a. As diferenças entre esses

dois padrões residem no fato de a largura de banda no 802.11p ser de 10 MHz em vez de 20 MHz e em algumas mudanças nas potências de transmissão e nas frequências de operação.

3. Protocolos de Disseminação de Dados

Muitos protocolos para realizar a disseminação de dados *broadcast* em redes veiculares foram propostos na literatura. As propostas atuais para disseminação de dados focam principalmente na escolha de quais veículos devem repassar os pacotes de dados de forma a garantir uma alta taxa de entrega de dados, baixa redundância nas transmissões e consequentemente, um baixo número de colisões [Ros et al. 2012, Viriyasitavat et al. 2010]. Para isso, tais protocolos calculam tempos de espera para a transmissão de pacotes de forma a dessincronizar as transmissões entre veículos vizinhos. Assim, veículos com tempos de espera menores, transmitem primeiro, suprimindo as transmissões dos demais veículos na vizinhança. As formas com que os tempos de espera são calculados podem ser baseadas em métodos estatísticos, topologia local, distância entre veículos, etc. A seguir, apresenta-se dois protocolos que utilizam diferentes abordagens.

Acknowledged Broadcast from Static to highly Mobile (ABSM) [Ros et al. 2012] é uma abordagem que utiliza o conceito de conjunto dominante mínimo (MCDS). ABSM baseia-se no fato de que o MCDS provê a melhor solução para o problema de disseminação de dados *broadcast* em uma rede conectada. O MCDS é o menor subconjunto de veículos conectados em que, se um veículo não estiver no MCDS, então ele está diretamente conectado a algum veículo que está no MCDS. Portanto, se todos os veículos no MCDS retransmitirem a mensagem, então todos os veículos na rede serão cobertos. No entanto, calcular o MCDS é um problem NP-difícil. Logo, ABSM emprega uma heurística que utiliza informações de vizinhos de um salto para determinar quais veículos estão no MCDS. Veículos no MCDS possuem maior prioridade para retransmitir as mensagens. Além disso, ABSM utiliza *beacons* periódicos como um mecanismo de reconhecimento implícito de recebimento das mensagens disseminadas, de maneira a garantir a entrega de dados em redes intermitentemente conectadas. Logo, no ABSM, quando um veículo recebe uma mensagem, ele espera pelo reconhecimento implícito dos vizinhos para então calcular o atraso para retransmitir a mensagem. Logo, a latência na entrega de mensagens depende da frequência com que os *beacons* são enviados.

An Urban Vehicular Broadcast Protocol (UV-CAST) [Viriyasitavat et al. 2010] é uma abordagem para disseminação de dados em centros urbanos. Nessa abordagem, quando um veículo recebe uma nova mensagem, utiliza-se informações de vizinhos de um salto para determinar se ele deve operar em um estado de supressão de *broadcast* ou *store-carry-forward*. Se o veículo determinar que ele deve ir para o estado de supressão de *broadcast*, então ele utiliza informações de mapa para verificar se está em uma interseção ou não de forma a calcular o atraso para retransmitir. Por outro lado, se o veículo perceber que ele deve operar em um estado de *store-carry-forward*, então ele verifica se é um veículo de borda ou não. O protocolo assume que veículos de borda possuem uma maior probabilidade de encontrar novos vizinhos. Logo, esses veículos armazenam e carregam a mensagem até que eles encontrem algum vizinho que não a tenha recebido. UV-CAST também utiliza reconhecimento implícito por intermédio de *beacons* para identificar os veículos que não receberam as mensagens disseminadas. Perceba que, no UV-CAST, apenas os veículos de borda são os responsáveis por armazenar, carregar e subsequente-

mente repassar as mensagens para os outros veículos. Além disso, quando um veículo recebe um *beacon* de um vizinho que não reconheceu o recebimento de uma determinada mensagem, então ele imediatamente retransmite a mensagem sem qualquer tipo de coordenação com os demais veículos na vizinhança. Essa falta de coordenação causa um aumento significativo no número de retransmissões, especialmente em cenários densos, conforme será mostrado na seção de avaliação de desempenho.

4. Impacto do padrão IEEE 802.11p nos protocolos de disseminação

Apesar dos protocolos de disseminação dessincronizarem as transmissões dos veículos em uma dada vizinhança, o mecanismo de salto de canais empregado pelo WAVE introduz um efeito indesejável de ressincronização nessas transmissões. No padrão de comunicação WAVE, a cada T_c segundos é permitido ao rádio saltar do CCH para o SCH e então, após mais T_c segundos, saltar do SCH para o CCH, e assim por diante. O padrão estabelece $T_c = 50$ ms. Diante disso, quando a camada MAC recebe da camada acima uma mensagem para ser transmitida no SCH, mas o CCH está momentaneamente ativo, então a mensagem deve aguardar o SCH se tornar ativo para que a transmissão possa ocorrer. No começo da operação de cada canal existe um intervalo de 5 ms chamado *guard interval* em que o canal é tratado como ocupado. Além disso, assume-se que os *beacons* periódicos, normalmente utilizados pelas aplicações de segurança, irão utilizar o CCH, enquanto as demais aplicações irão utilizar o SCH. Toda mensagem repassada para a camada MAC deve especificar em qual canal ela deve ser transmitida.

Essa abordagem de salto de canais introduz um efeito de sincronização indesejado. Considere o exemplo mostrado na Figura 2. Nessa figura, há dois veículos tentando retransmitir uma mensagem que eles receberam de um vizinho em comum. No tempo T_1, ambos os veículos agendam o repasse da mensagem para a camada MAC. Perceba que, o veículo A calcula e agenda o repasse com um atraso de 10 ms, enquanto B calcula e agenda o repasse com um atraso de 25 ms. Sem o mecanismo de salto de canais, A iria transmitir primeiro, B iria escutar a transmissão de A e, consequentemente, iria cancelar sua própria transmissão, evitando assim uma transmissão desnecessária. No entanto, não é o que ocorre quando a abordagem de salto de canais é empregada. No tempo T_2, ou seja, 10 ms depois de T_1, o temporizador *rebroadcast_timer* em A expira e ele repassa a mensagem para a camada MAC para transmissão no SCH. Porém, perceba que o CCH está momentaneamente ativo. Logo, a camada MAC do veículo A armazena a mensagem até que o SCH fique ativo. No tempo T_3, ou seja, 25 ms depois de T_1, é o momento em que o temporizador *rebroadcast_timer* em B expira e então a mensagem é repassada para a camada MAC, também para transmissão no SCH. Assim como em A, a camada MAC do veículo B também armazena a mensagem, já que o CCH está momentaneamente ativo. Finalmente, no tempo T_4, quando o SCH se torna ativo, a camada MAC de ambos os veículos escuta e percebe o canal como não ocupado. Assim, os veículos transmitem a mensagem ao mesmo tempo, resultando em uma colisão. Perceba que, apesar do fato do algoritmo de supressão de *broadcast* ter realizado seu devido trabalho em calcular valores diferentes para o atraso de retransmissão para os veículos (dessincronização), ao final, as retransmissões ocorreram ao mesmo tempo (ressincronização). Diante disso, argumentamos que a utilização apenas de mecanismos de supressão de *broadcast* não é suficiente para evitar o problema da tempestade de *broadcast* em redes VANETs densas, especialmente na transmissão de dados em rajadas [Eckhoff 2012]

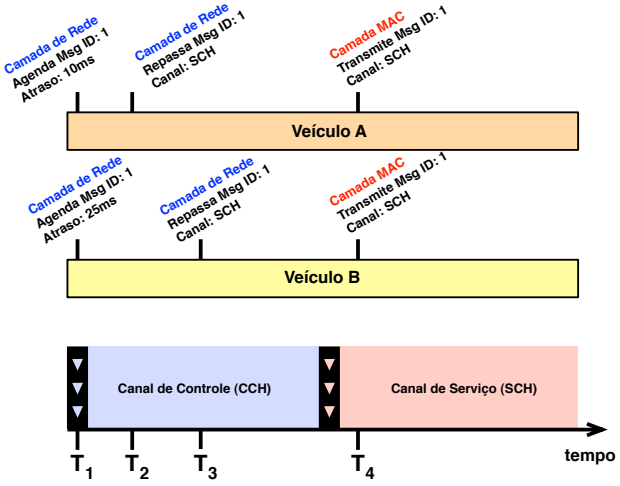


Figura 2. Efeito de sincronização introduzido pelo mecanismo de salto de canais utilizado na camada MAC do IEEE 802.11p

Além do efeito de sincronização indesejado introduzido, outro fator importante é a forma como as categorias de acesso influenciam no desempenho dos protocolos. À medida que a carga da rede aumenta, as categorias de acesso com menor prioridade, AC0 e AC1, conseguem um menor acesso ao meio e, portanto, há uma diminuição da taxa de pacotes enviados a partir destas categorias de acesso [Eichler 2007]. Outro parâmetro importante que ilustra o impacto das categorias de acesso no desempenho é o atraso fim-a-fim, onde, a medida que a densidade da rede aumenta, as categorias de menor prioridade apresentam um maior crescimento no atraso, especiamente a AC0 que sofre um atraso exponencial com o crescimento da densidade da rede [Eichler 2007].

5. Solução Proposta

De forma a evitar o problema supracitado, propõe-se um mecanismo que recebe o atraso para retransmitir T calculado por qualquer protocolo de disseminação de dados e, caso necessário, recalcula um novo atraso para retransmitir T_d de acordo com o regime de salto de canais. O Algoritmo 1 mostra como tal mecanismo funciona. Assume-se que o novo atraso T_d, calculado por nossa solução, é para uma mensagem a ser transmitida no SCH. No entanto, o algoritmo pode ser facilmente estendido de forma a calcular o atraso para mensagens a serem transmitidas no CCH. A nossa solução pode ser acoplada em qualquer protocolo de disseminação de dados. Nos protocolos para realizar a disseminação de dados em redes veiculares propostos na literatura e avaliados neste trabalho, um dado veículo, após calcular o atraso T para retransmitir uma mensagem m (ou seja, atraso para repassar a mensagem para a camada MAC para que ocorra a transmissão), chama o mecanismo proposto *RAN*, a qual retorna um novo valor para o atraso (T_d).

A idéia geral do Algoritmo 1 é adicionar ao atraso original T para retransmitir uma mensagem m, um atraso extra T_c, caso necessário. Para este trabalho, foi definido que tal atraso extra T_c é o intervalo de tempo atribuído para cada *slot* (canal) de transmissão.

No entanto, o novo atraso T_d somente seria calculado no caso de haver outros veículos querendo transmitir no mesmo *slot* de transmissão (canal), como ilustrado na

Algorithm 1: RAN: **R**esynchronization **A**ware of the **N**eighbourhood

```
1  T ← atraso para retransmitir que é passado para a função;
2  T_c ← tempo para cada canal (50 ms);
3  T_s ← tempo restante para que ocorra uma mudança de canal;
4  se Vizinhostransmitindo então
5      retorna T;
6  atrasoAux ← T − T_s;
7  se strasoAux < 0 então
8      atrasoAux ← 0;
9  ciclos ← ⌊atrasoAux / T_c⌋;
10 se CCH está ativo então
11     se ciclos%2 == 1 então
12         T_d ← T + T_c ;
13 senão
14     se ciclos%2 == 0 então
15         T_d ← T + T_c ;
16 retorna T_d;
```

Figura 2. Na Figura 2, dois veículos agendam a transmissão para o mesmo *slot* (canal de controle). Para isso, todos os veículos enviam, através das mensagens de *beacon*, o código e o tempo programado para o envio de todas as suas mensagens programadas. Os nós vizinhos recebem os *beacons* e armazenam a tabela de todos os outros vizinhos. Logo, quando um veículo pretende enviar uma mensagem m, primeiramente, este verifica na tabela se existe algum vizinho com mensagem programada para o intervalo em que m está inicialmente programada, ou seja, utilizando o atraso de propagação T. Caso exista, o cálculo do novo atraso de propagação T_d é calculado, caso contrário, o atraso T é retornado sem alterações, pois não haverá concorrência de acesso ao meio no canal pretendido.

Como exemplo do funcionamento do mecanismo, na Figura 3, um veículo recebe a mensagem m no tempo T_1. Assuma, neste exemplo, que existe concorrência para transmitir no meio, no mesmo intervalo de tempo, ou seja, o novo atraso de propagação é calculado. Neste caso, o protocolo utilizado para realizar a disseminação de dados calcula um atraso $T = 70$ ms para retransmitir a mensagem m. Perceba que T_c = 50 ms e, que no tempo T_1, o CCH está momentaneamente ativo e assim permanecerá por mais $T_s = 5$ ms. Neste exemplo, o algoritmo 1 calcula o número de ciclos de canais inteiros antes de T, ou seja, $cicles$ = 1. Este número é necesário para descobrir se o CCH está ativo após o atraso de propagação T. No primeiro caso, se a camada de rede envia m para a camada MAC no tempo T_1 e esta espera T = 70 ms antes de tentar transmitir m no SCH, então m perceberá que o CCH esta ativo e ainda permanecerá por 40 ms. Note que se a camada de rede não utilizar o mecanismo de dessicronização proposto aqui, esta agenda a retransmissão de m com o atraso de propagação T. Assim, no tempo T_2, ela envia m para a camada MAC para transmissão no SCH. No entanto, a transmissão não pode ocorrer prontamente, pois o CCH está ativo, o que pode levar a uma ressincronização, conforme mostrado anteriormente. Por outro lado, se a camada de rede utilizar o Algoritmo 1, então ela agenda

a retransmissão de m com um atraso $T_d = T + T_c$. Assim, no tempo T_3, a camada de rede repassa m para a camada MAC e esta pode imediatamente efetuar a transmissão, já que o SCH está momentaneamente ativo. De fato, a utilização do atraso para retransmitir $T_d = T + T_c$ garante que quando a camada de rede repassa m para a camada MAC, então o SCH sempre estará ativo, consequentemente evitando o efeito de ressincronização do atraso.

Em resumo, utilizar o Algoritmo 1 produz exatamente o mesmo resultado que utilizar o atraso T e um relógio que funciona somente quando o SCH está ativo, ou seja, o tempo avança somente quando SCH está ativo.

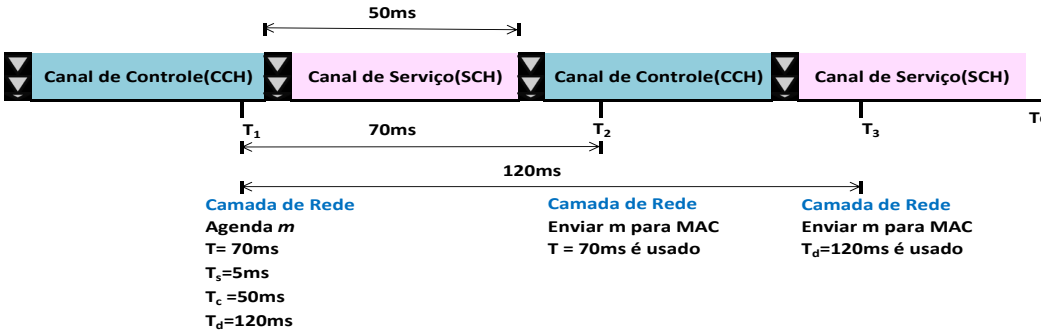


Figura 3. Exemplo que mostra como o mecanismo de dessincronização funciona

6. Avaliação de desempenho

Nesta seção, apresentamos a avaliação de desempenho do mecanismo *RAN*. Para isso, foi feita uma série de simulações usando o simulador OMNeT++ 4.2.2 [Varga 2008], onde comparamos o uso do *RAN* em dois protocolos de disseminação de dados: UV-CAST e ABSM. Estes foram escolhidos para serem algoritmos de disseminação atuais e bem aceitos na comunidade científica.

6.1. Metodologia

O mecanismo proposto *RAN* foi implementado em dois protocolos para disseminação de dados propostos na literatura, sendo eles ABSM e UV-CAST. Os protocolos com o mecanismo *RAN* são nomeados como ABSM-RAN e UV-CAST-RAN. Avaliamos a performance dos protocolos utilizando o cenário em *grid Manhattan*. Este cenário é composto por dez ruas de pista dupla, uniformemente espaçadas, nos sentidos vertical e horizontal, numa área de 1 km^2. Neste ambiente, foram avaliados dois cenários, no primeiro, variamos a densidade do tráfego de veículos e, no segundo, variamos a categoria de acesso na camada MAC do padrão WAVE.

Para simular a mobilidade de veículos utilizamos o *SUMO* 0.17.0 *mobility simulator* [Behrisch et al. 2011]. Um veículo situado, aproximadamente, no centro do cenário gera 100 mensagens de 1024 *bytes*, que são disseminadas utilizando uma taxa de 1 Mbit/s para todos os outros veículos na rede. Além disso, usamos o *Veins* 2.1 [Sommer 2011] para modelar a rede de forma a aumentar a precisão dos resultados. Como parâmetros, configuramos a potência de transmissão em 0.98 mW, resultando num alcance de transmissão de aproximadamente 200 metros. Além disso, os resultados representam a média de 33 execuções para cada cenário com um intervalo de confiança de 95%.

As métricas utilizadas neste trabalho são:

- **Taxa de entrega**: porcentagem de mensagens de dados geradas pelo veículo fonte que é recebida pelos veículos pretendidos. A expectativa é que o protocolo de disseminação consiga entregar 100% das mensagens enviadas.
- **Atraso**: tempo médio gasto para a mensagem de dados viajar da fonte para o destino.
- **Colisões**: média do número de colisões por veículo para disseminar todas as mensagens de dados. O alto número de colisões indica que um dado protocolo não é capaz de evitar o problema de *broadcast storm*.

6.2. Cenario 1 – variando a densidade do tráfego de veículos

Este cenário foi definido para observar o comportamento dos protocolos de acordo com a variação da densidade de veículos por km^2. Inicialmente, foi aplicado o número de 200 veículos por km^2 e chegamos ao valor de 600 veículos por km^2. Isso permite avaliar o mecanismo *RAN* em diferentes condições de tráfego de veículos. A Tabela 1 mostra os parâmetros de simualação específicos para este cenário.

Tabela 1. Parâmetros do Cenário 1

Parâmetro	Valor
Densidade	200, 300, 400, 500, 600 veículos/km^2
Categoria de Acesso	0
Taxa de Transmissão	1 Mbit/s
Tamanho da Mensagem de Dados	1024 bytes
Número de mensagens Produzidas	100

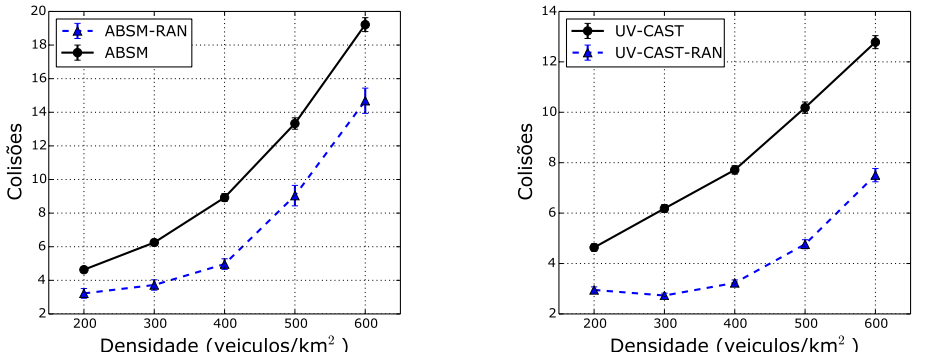
6.2.1. Colisão

As Figuras 4(a) e 4(b) mostram o número de colisões por veículo ocorridas para os dois protocolos ABSM e UV-CAST, variando a densidade do tráfego de veículos. A Figura 4(a) ilustra o comportamento do protocolo ABSM em dois modos, com e sem a utilização do mecanismo *RAN*, respectivamente, ABSM-RAN e ABSM. Da mesma maneira a Figura 4(b) para o protocolo UV-CAST.

Resultados de simulação mostram que os protocolos que fazem uso do mecanismo proposto *RAN* apresentam menor número de colisões. Na média, o ABSM apresenta 10.4 colisões por veículo, enquanto que o ABSM-RAN obteve 6.5 colisões por veículo, ou seja, uma redução de aproximadamente 37% de colisões. No caso do UV-CAST, ilustrado na Figura 4(b), o UV-CAST tem 8.1 colisões por veículo, enquanto que o UV-CAST-RAN obteve 4.2 colisões por veículo, ou seja, uma redução de aproximadamente 49% de colisões. Isso mostra quanto o mecanismo proposto *RAN* pode melhorar os protocolos avaliados evitando o número de colisões, um dos principais objetivos do mecanismo.

6.2.2. Taxa de Entrega

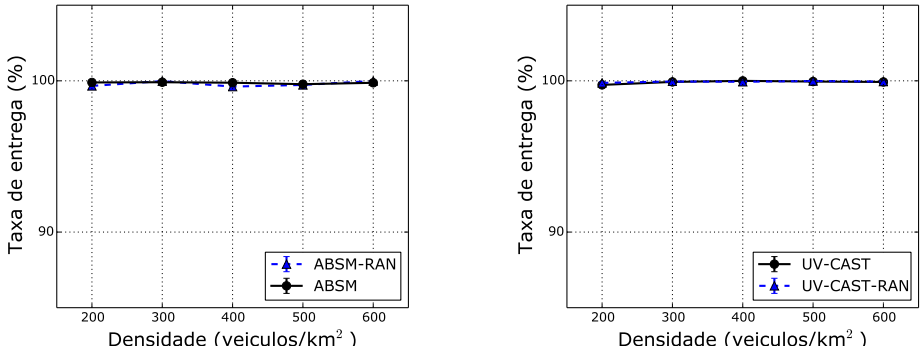
As Figuras 5(a) e 5(b) mostram a taxa de entrega de pacotes para os dois protocolos ABSM e UV-CAST, variando a densidade do tráfego de veículos. Os protocolos que fa-



(a) Número de colisões – ABSM (b) Número de colisões – UV-CAST

Figura 4. Número de colisões por veículo.

zem uso do mecanismo proposto *RAN* apresentam praticamente a mesma taxa de entrega em relação ao protocolo original. Na média, para os quatro casos, ABSM, ABSM-RAN, UV-CAST e UV-CAST-RAN, a taxa de entrega é bem próxima de 100%. Esses números ilustram que a utilização do mecanismo *RAN* não afeta a taxa de entrega dos protocolos originais, ABSM e UV-CAST. Vale ressaltar que a taxa de entrega é calculada através do recebimento do pacote de dados pelo veículo de destino independentemente de quantas vezes o veículos de origem enviou. Por isso, mesmo sem a utilização do mecanismo, a taxa de entrega é aproximadamente 100% e com a utilização do mecanismo há um pequeno aumento nos mesmos, porém com uma diminuição do número de colisões.



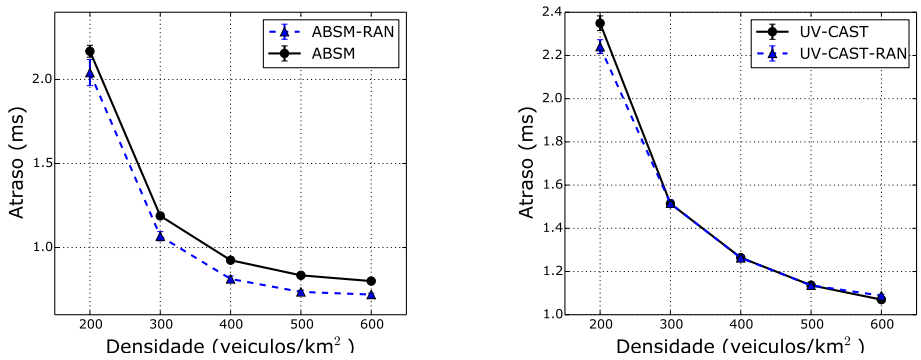
(a) Taxa de Entrega – ABSM (b) Taxa de Entrega – UV-CAST

Figura 5. Taxa de entrega.

6.2.3. Atraso

As Figuras 6(a) e 6(b) mostram o atraso médio das mensagens no envio desde a origem até o destino para os dois protocolos ABSM e UV-CAST variando a densidade do tráfego de veículos. Ambas as figuras mostram que os resultados são muito similares. O ABSM-RAN apresenta uma pequena vantagem sobre o ABSM. No caso do UV-CAST, como mostra a Figura 5(b), UV-CAST e UV-CAST-RAN apresentam quase os mesmos resultados. Isto mostra que o atraso introduzido pelo mecanismo *RAN* não afetou o atrado

médio das menssagens. Este resultado é devido à redução no número de colisões, pois estas requerem uma retransmissão, consequentemente, um aumento no atraso (T) para a retransmissão.



(a) Atraso no ABSM (b) Atraso no UV-CAST

Figura 6. Atraso.

Conforme a Tabela 1, utilizamos a categoria de acesso igual a 0. Entre as categorias de acesso, essa foi a que o *RAN* apresentou os piores resultados. Isso é apresentado, em detalhes, na próxima seção.

6.3. Cenario 2 – variando a categoria de acesso na camada MAC do padrão WAVE

Neste cenário, o objetivo principal é observar o comportamento dos protocolos de acordo com a variação da categoria de acesso na camada MAC do padrão WAVE. Como citado na Seção 2, os possíveis valores são: 0, 1, 2 e 3. A Tabela 2 mostra os parâmetros de simulação específicos usados.

Tabela 2. Parâmetros do Cenário 2

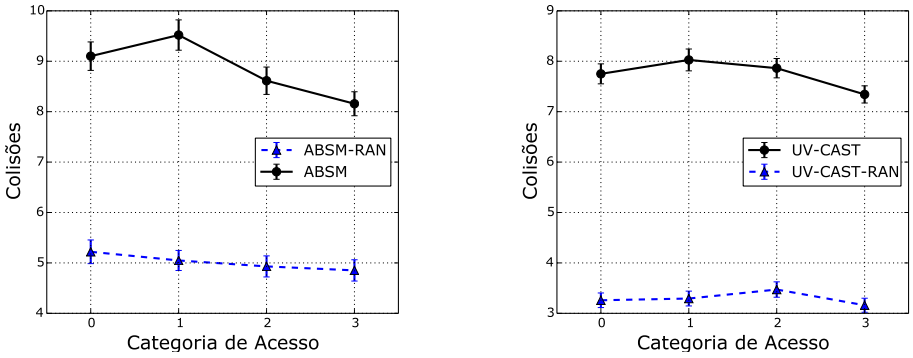
Parameter	Value
Categoria de Acesso	0, 1, 2, 3
% Densidade	400 veículos/km^2
Taxa de Transmissão	1 Mbit/s
Tamanho da Mensagem de Dados	1024 bytes
Número de mensagens Produzidas	100

6.3.1. Colisão

As Figuras 7(a) e 7(b) mostram o número de colisões por veículo ocorridas para os dois protocolos ABSM e UV-CAST, variando a categoria de acesso e mantendo constante a densidade do tráfego de veículos, no caso, em 400 veículos/km^2. Resultados de simulação mostram que os protocolos que fazem uso do mecanismo proposto *RAN* apresentam menor número de colisões.

Na média, o ABSM apresenta 8.8 colisões por veículo, enquanto que o ABSM-RAN obteve 5.1 colisões por veículo (veja a Figura 7(a)), ou seja, uma redução de apro-

ximadamente 42% de colisões. No caso do UV-CAST, ilustrado na Figura 7(b), o UV-CAST tem 7.7 colisões por veículo, enquanto que o UV-CAST-RAN obteve 3.25 colisões por veículo, ou seja, uma redução de aproximadamente 57% de colisões. Isso mostra quanto o mecanismo proposto *RAN* pode melhorar os protocolos avaliados evitando o número de colisões, um dos principais objetivos do mecanismo.



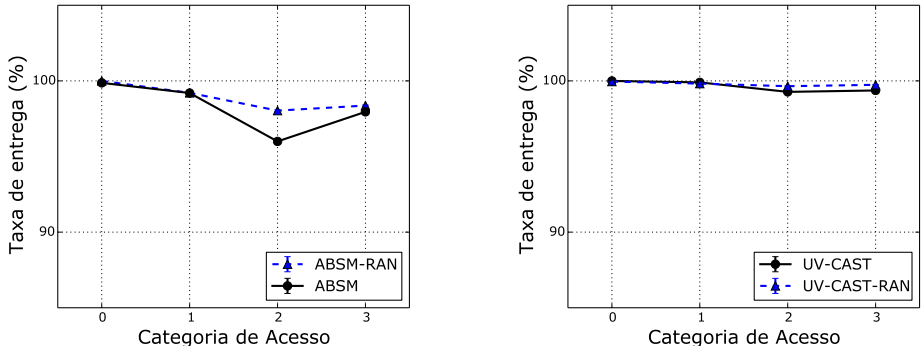
(a) Número de colisões por veículo no ABSM

(b) Número de colisões por veículo no UV-CAST

Figura 7. Número de colisões por veículo.

6.3.2. Taxa de Entrega

As Figuras 8(a) e 8(b) mostram a taxa de entrega de pacotes para os dois protocolos ABSM e UV-CAST, variando a categoria de acesso conforme a Tabela 2. Na média, os protocolos que fazem uso do mecanismo proposto *RAN* apresentam maior taxa de entrega em relação ao protocolo original. O ABSM-RAN apresenta melhora média de 1.5% (veja a Figura 8(a)). No caso do UV-CAST, mostrado na Figura 8(b), a taxa de entrega é bem próxima de 100%, apesar do UV-CAST-RAN apresentar uma pequena vantagem. Esses números são esperados devido ao número de colisões, conforme apresentado na seção anterior, os protocolos originais apresentam um número maior de colisões.



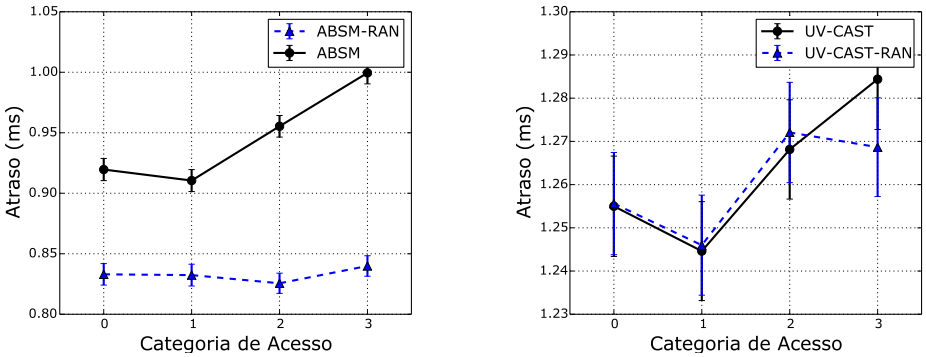
(a) Taxa de Entrega no ABSM

(b) Taxa de Entrega no UV-CAST

Figura 8. Taxa de entrega.

6.3.3. Atraso

Como esperado, os protocolos que fazem uso do mecanismo proposto *RAN* apresentam um menor atraso (Veja as Figuras 9(a) e 9(b)). Entre as categorias de acesso, a categoria de acesso 0 foi a que apresentou o pior resultado, como mencionado anteriormente. Isto acontece porque, entre as categorias de acesso, a categoria 0 possui o maior tempo de acesso ao meio e a categoria de acesso 3 possui o maior tempo de acesso ao meio. Quanto menor tempo de acesso ao meio, maior o número de colisões e assim o mecanismo *RAN* apresenta melhores resultados, isto é, o mecanismo torna o protocolo menos vulnerável para o problema. Na média, o ABSM-RAN apresenta um melhoramento de 12%. No caso do UV-CAST e UV-CAST-RAN apresentam praticamente o mesmo resultado, porém, o UV-CAST-RAN apresenta melhor resultado na categoria de acesso 3, ou seja, quando a categoria de acesso apresenta o menor tempo de acesso ao meio.



(a) Atraso no ABSM (b) Atraso no UV-CAST

Figura 9. Atraso.

7. Conclusão

Neste trabalho, foi proposto o *RAN*, um mecanismo de dessincronização na camada MAC do padrão IEEE 802.11p/WAVE. Foram utilizados dois protocolos – ABSM e UV-CAST – para avaliar o desempenho do mecanismo proposto *RAN*. Para cada protocolo citado, foi avaliado o desempenho do protocolo original com uma versão que implementa o *RAN*. Os resultados das simulações mostraram que os protocolos que utilizam o mecanismo *RAN* apresentaram um melhor desempenho considerando as métricas de colisão e taxa de entrega. Como trabalhos futuros, pretendemos melhorar o mecanismo afim de reduzir o atraso introduzido pela sua utilização. Também pretendemos testar o *RAN* em outros protocolos similares.

Acknowledgment

Os autores gostariam de agradecer o apoio financeiro concedido da CAPES e CNPq para o bolsista de doutorado Erick Aguiar Donato. Ainda, os autores também agradecem ao CNPq por financiar partes dos seus projetos de pesquisas. Por fim, Leandro Villas agradece o apoio financeiro da FAPESP por meio do processo nº 2015/07538-1, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências

Barradi, M. e Hafid, A. e. G. J. (2010). Establishing Strict Priorities in IEEE 802.11p WAVE Vehicular Networks. In *IEEE Global Telecommunications Conference (GLOBECOM '10)*, pages 1–6.

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). SUMO - Simulation of Urban MObility: An Overview. In *International Conference on Advances in System Simulation (SIMUL '11)*, pages 63–68.

Campolo, C. e Molinaro, A. e. V. A. (2011). Understanding the performance of short-lived control broadcast packets in 802.11p/WAVE Vehicular networks. In *IEEE Vehicular Networking Conference (VNC '11)*, pages 102–108.

Eckhoff, D. e Sommer, C. e. D. F. (2012). On the Necessity of Accurate IEEE 802.11p Models for IVC Protocol Simulation. In *IEEE Vehicular Technology Conference (VTC '12)*, pages 1–5.

Eichler, S. (2007). Performance Evaluation of the IEEE 802.11p WAVE Communication Standard. In *IEEE Vehicular Technology Conference (VTC '07)*, pages 2199–2203.

Grafling, S. e Mahonen, P. e. R. J. (2010). Performance evaluation of IEEE 1609 WAVE and IEEE 802.11p for vehicular communications. In *International Conference on Ubiquitous and Future Networks (ICUFN '10)*, pages 344–348.

Hartenstein, H. e Laberteaux, K. P. (2008). A tutorial survey on vehicular ad hoc networks. *IEEE Communications Magazine*, 46(6):164–171.

IEEE (2010). Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. IEEE Standards.

Ros, F., Ruiz, P., and Stojmenovic, I. (2012). Acknowledgment-Based Broadcast Protocol for Reliable and Efficient Data Dissemination in Vehicular Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 11(1):33–46.

Sommer, Christoph e German, R. e. D. F. (2011). Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15.

Varga, András e Hornig, R. (2008). An overview of the OMNeT++ simulation environment. In *International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (Simutools '08)*, pages 1–10.

Villas, L. A. e Boukerche, A. e. M. G. e. P. R. W. e. L. A. A. F. (2014). Drive: An efficient and robust data dissemination protocol for highway and urban vehicular ad hoc networks. *Computer Networks*, 75, Part A(0):381 – 394.

Viriyasitavat, W., Bai, F., and Tonguz, O. K. (2010). Uv-cast: An urban vehicular broadcast protocol. In *Vehicular Networking Conference (VNC), 2010 IEEE*, pages 25–32.

Trilha Principal do SBRC 2016
Sessão Técnica 7
Computação na Nuvem

Sincronização de Arquivos entre Nuvens de Armazenamento e Repositórios Geograficamente Distribuídos

Gil Andriani, Guilherme Koslovski, Mauricio A. Pillon

[1]Programa de Pós Graduação em Computação Aplicada (PPGCA)
Universidade do Estado de Santa Catarina - UDESC - Joinville, SC - Brasil

gil.andriani@gmail.com, {guilherme.koslovski, mauricio.pillon}@udesc.br

Resumo. *Provedores de nuvens computacionais difundiram sistemas de armazenamento dinâmicos, entregues aos usuários finais como serviços acessíveis sob-demanda. Especificamente, o armazenamento de arquivos alcançou popularidade em ambientes domésticos. Entretanto, os requisitos de acesso, as expectativas de desempenho e as características de utilização são distintas para organizações, fato não considerado pelos aplicativos populares para sincronização de arquivos entre provedores e repositórios locais. Nesse contexto, esse artigo apresenta CHSAN, uma arquitetura para sincronização de arquivos entre armazenamento na nuvem e repositórios geograficamente distribuídos. Os resultados experimentais indicam uma aplicação promissora em ambientes com usuários colaborativos geograficamente distribuídos.*

Abstract. *Cloud Computing providers had spread the dynamic storage provisioning, delivered to end users as on-demand services. Although file storage and share achieved popularity among domestic users, access requirements, performance expectations and usage characteristics are different for organizations, and were not originally considered by popular application for synchronizing files between providers and local repositories. In this context, this paper introduces CHSAN, an architecture for synchronizing files between cloud storage and repositories geographically distributed. The experimental results indicate a promising application in collaborative environments with geographically distributed users.*

1. Introdução

As nuvens computacionais revolucionaram a utilização de recursos de processamento, comunicação e armazenamento, entregando essas funcionalidades aos usuários como serviços sob-demanda [Zhang et al. 2010]. A adoção de armazenamento na nuvem, sobretudo, é uma realidade para usuários domésticos. O armazenamento, a edição e o compartilhamento de arquivos foram popularizados pela difusão de clientes para sincronização (*e.g.*, DropBox[1], Google Drive[2] e OneDrive[3]) de arquivos com repositórios hospedados em nuvens computacionais.

O armazenamento na nuvem fornece aos seus usuários maior disponibilidade aos dados [Drago et al. 2012], entretanto, essas ferramentas foram concebidas com foco em usuários domésticos. Gonçalves [Gonçalves et al. 2014] identificou o perfil destes usuários

[1]DropBox, disponível em `https://www.dropbox.com/`.
[2]Google Drive, disponível em `https://www.google.com/drive/`.
[3]Microsoft OneDrive, disponível em `https://onedrive.live.com/`.

em campi, apontando o tamanho médio dos repositórios como $4,23$ GB, sendo que a maioria dos usuários tende a armazenar muitos arquivos (mais de 1.000 arquivos), incluindo usuários com mais de 20.000 arquivos armazenados. Referente à dinâmica de atualização, foi observado que cerca de 82% das operações de atualização carregam até 1 MB, caracterizando uma concentração de uso sobre arquivos pequenos. Ainda, a maioria das sessões de conexão entre aplicativos clientes e provedores (85% das sessões) não tem nenhuma transferência significativa de arquivos.

Embora populares, os clientes de sincronização de arquivos na nuvem não atendem aos requisitos específicos de organizações com múltiplos usuários colaborativos geograficamente distribuídos, concentrados em redes privadas e sistemas de armazenamento legados [Smith et al. 2013][Drago et al. 2012]. Diferente dos usuários domésticos, em organizações ocorre a interação em projetos: diversos usuários atuam em arquivos compartilhados, como planilhas de dados, sistemas de armazenamento e gerenciamento de tarefas empresariais. Os grupos de usuários colaborativos são usualmente dinâmicos e temporários, formados espontaneamente segundo interesses profissionais ou pessoais. Organizações com múltiplos usuários colaborativos geograficamente distribuídos tendem a formar grupos colaborativos concentrados em redes privadas e geograficamente distribuídas. É comum a existência de uma conexão dedicada e privada para tráfego de dados sobre redes metropolitanas.

Quanto aos sistemas de armazenamento legados, tradicionalmente, organizações com múltiplos usuários dispõem de tecnologias acessíveis em redes privadas, desenvolvidas sobre soluções *Network File System (NFS)*, *Common Internet File System (CIFS)* ou *Server Message Block (SMB)*. A existência destes recursos é fator limitante para a migração de soluções para a nuvem, sendo parcialmente responsável pelo fato de 45% das organizações não terem previsão para migração de qualquer serviço para nuvens computacionais [Smith et al. 2013]. A latência no acesso aos arquivos é um ponto crítico para a utilização do sistema.

As ferramentas de sincronização mantém o conteúdo do usuário sincronizado com o armazenamento na nuvem, realizando cópias sincronizadas do conteúdo compartilhado a cada acesso para cada dispositivo. Este não é um fator limitante para os usuários domésticos, pois usualmente a capacidade de seu espaço virtual é semelhante (ou inferior) ao espaço em seus dispositivos pessoais. Entretanto, o conteúdo compartilhado entre equipes de uma organização facilmente supera o espaço individual das estações de trabalho. Os repositórios de dados disponíveis nestas organizações, normalmente, possuem capacidade de armazenamento superior a capacidade do disco local das estações de trabalho [Douceur and Bolosky 1999]. No caso do armazenamento em nuvem, o acesso à totalidade dos dados fica restrito à disponibilidade de armazenamento do dispositivo com menor capacidade. Assim, a colaboração entre os usuários do grupo depende dos arquivos selecionados para sincronização, atualmente, sob responsabilidade do próprio usuário. Quando colaborando, os usuários devem reconfigurar os repositórios locais.

Ainda, o amplo acesso à rede através da mobilidade e da concentração de profissionais em um mesmo local, são fatores conflitantes. Por um lado, a disponibilidade de acesso aos dados de qualquer lugar permite maior interação e colaboração entre as equipes, por outro lado, a concentração de profissionais em um mesmo local acessando o mesmo volume de dados na nuvem pode sobrecarregar os pontos de acesso à Internet das organizações, além de aumentar o tempo de acesso aos arquivos.

Considerando a lacuna quanto à manutenção das funcionalidades dos sistemas de armazenamento legados e as soluções para compartilhamento de arquivos na nuvem que impedem parcialmente a ampla utilização das nuvens computacionais por parte organizações, este trabalho apresenta a arquitetura do CHSAN (Cliente Híbrido para Sincronização de Arquivos em Nuvem). A arquitetura proposta aplica técnicas já consolidadas em armazenamento distribuído no contexto de sincronização com nuvens de armazenamento. Esta abordagem híbrida contorna as limitações de capacidade com o uso de armazenamento local, hierárquico e seletivo. O limite de escalabilidade é atenuado com a introdução de um mecanismo que reduz o número de consultas ao repositório da nuvem. Ainda, CHSAN permite a indexação de todo o sistema de arquivo da nuvem. O arquivo pode estar no disco local, em um sistema distribuído na rede interna ou na nuvem. A análise experimental concentrou-se em dois cenários, (i) avaliação do impacto no desempenho de operações sobre arquivos no acesso ao repositório de metadados e (ii) avaliação do desempenho das operações em arquivos compartilhados localizados unicamente na nuvem. Embora a arquitetura acrescente módulos para a combinação de repositórios locais e na nuvem, o tempo necessário para sincronização não foi significativamente acrescido quando comparado a soluções existentes.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 descreve o problema relacionado com a sincronização de arquivos. A arquitetura CHSAN é apresentada na Seção 3, enquanto a análise experimental é discutida na Seção 4. Os trabalhos relacionados são revisados na Seção 5, e finalmente na Seção 6 apresenta as considerações finais e perspectivas para trabalhos futuros.

2. Sincronização de Arquivos em Nuvens: Conceitos e Problemática

O cenário de nuvens computacionais, armazenamento e compartilhamento de arquivos, compreende dois atores principais: os repositórios de armazenamento e os clientes de sincronização. O repositório é usualmente composto por módulos para notificação, indexação e interface de operação, sendo implementado como um sistema de arquivos distribuídos que assegure disponibilidade, autenticidade, confidencialidade e posse. É comum a disponibilização de serviços para controle de versionamento, quotas de compartilhamento e edição colaborativa.

Por sua vez, um cliente de sincronização de arquivos é responsável por manter uma determinada estrutura de diretórios e de arquivos em sincronia com o repositório da nuvem. Em suma, o controle é realizado por sincronizadores e observadores, que atuam sobre os metadados relacionados com os arquivos. Existem dois observadores (local e nuvem), responsáveis por identificar alterações nos arquivos. Quando necessário, um módulo sincronizador realiza a transferência dos arquivos (parcial ou total) entre os repositórios. Algumas funcionalidades comuns são encontradas em clientes populares como sincronização incremental, criptografia e compressão de dados. Ainda, estratégias de otimização foram aplicadas para diminuir o volume de dados trafegados entre os repositórios de armazenamento e os clientes de sincronização (eg., DropBox LanSync[4]).

Há alguns anos, clientes de sincronização de arquivos na nuvem estão difundidos entre usuários domésticos, atendendo satisfatoriamente suas necessidades. Recentemente, provedores de armazenamento na nuvem passaram a oferecer serviços voltados para organizações, com maior capacidade de armazenamento do que os pacotes para usuários

[4]DropBox LanSync disponível em `https://www.dropbox.com/en/help/137`.

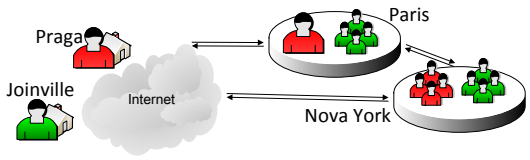


Figura 1. Ambiente organizacional com múltiplos usuários colaborativos.

domésticos [Naldi and Mastroeni 2013]. No entanto, ambientes organizacionais são constituídos por múltiplos usuários que colaboram em projetos comuns, muitas vezes em ambientes geograficamente distribuídos concentrados em redes privadas. Na Figura 1 tem-se um cenário, onde a organização descrita possui dois grupos: os usuários em vermelho e os em verde, situados fisicamente nas cidades de Nova York, Paris, Joinville e Praga. Esta organização tem dois escritórios físicos, localizados em Nova York e Paris, onde concentram-se o maior número de colaboradores que interagem por meio de redes privadas interligadas. Em Joinville e Praga encontram-se dois usuários geograficamente distantes dos escritórios que colaboram com seus grupos via Internet. Portanto, esta organização possui múltiplos usuários geograficamente distribuídos, concentrados em Nova York e Paris, conectados entre si via redes privadas e Internet, que colaboram pelo uso do compartilhamento de arquivos.

O compartilhamento de arquivos entre estes usuários pode ser efetuado via sistemas de armazenamento privado, desenvolvidos sobre soluções NFS, CIFS ou SMB. Estas soluções são usualmente empregadas em redes privadas locais ou ambientes geograficamente distribuídos, interconectados com enlaces dedicados. Os sistemas de armazenamento privado contemplariam a colaboração entre os usuários de Paris e de Nova York. No entanto, esta escolha não permitiria a integração dos usuários de Joinville e Praga, pois eles estão fora da rede privada com uma conexão Internet. Para estes usuários, uma alternativa viável remete ao armazenamento em nuvem computacional. O conteúdo compartilhado na nuvem é sincronizado nos discos locais de cada cliente. Para que a sincronização ocorra, é necessário que a quantidade de dados a ser copiada seja inferior a menor unidade de armazenamento envolvida: limitado em função do cliente ou limitado em função do espaço de armazenamento disponível na nuvem. Normalmente, essa limitação de capacidade encontra-se nas estações de trabalho. É comum a existência de organizações em que uma estação de trabalho possui unidades de armazenamento na ordem de centenas de gigabytes, enquanto o armazenamento em nuvem está na ordem de centenas de terabytes [Katzer and Crawford 2013].

Nos clientes populares, existe a possibilidade de escolher quais diretórios serão sincronizados. Entretanto, com múltiplos arquivos e usuários, a elaboração e manutenção de uma hierarquia de compartilhamento seletiva não é uma tarefa trivial. Ainda, a sincronização seletiva, da forma como foi implementada pelos clientes de sincronização, fere o amplo acesso difundido pelo armazenamento em nuvem [Mell and Grance 2011]: o recurso permanece disponível no repositório de armazenamento, mas as funcionalidades de gerenciamento e utilização locais não podem ser aplicadas.

Quanto à distribuição geográfica dos usuários, as organizações compartilham o mesmo ponto de acesso à Internet com diversos usuários internos. No cenário descrito na Figura 1 observa-se usuários concentrados em redes privadas interligadas nas cidades de Nova York e de Paris. Nesse cenário, a utilização do serviço pode ser limitada em função da quantidade de dados a ser escrito na nuvem frente à quantidade de clientes que neces-

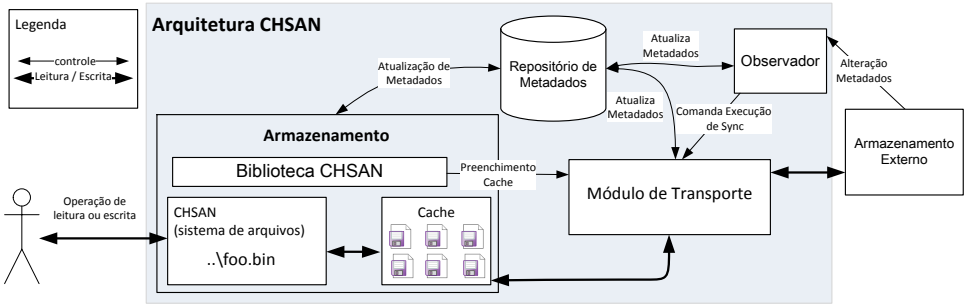


Figura 2. Arquitetura do cliente híbrido de sincronização de arquivos CHSAN.

sita compartilhar a mesma informação. Ainda, é comum que conjuntos de arquivos sejam compartilhados, e usualmente, existe um canal de comunicação dedicado para tráfego de informações sobre redes metropolitanas. A escolha do armazenamento na nuvem para este cenário, implicaria no aumento da latência no acesso aos arquivos compartilhados entre usuários de Paris e Nova York. A junção dessas características (modelo de sincronização explorado pelos clientes, concentração de usuários com os mesmos documentos compartilhados e existência de canais dedicados) faz com que as ferramentas de sincronização em nuvem, nestes ambientes, tenham percentuais de adoção inferiores aos observados com usuários domésticos [Smith et al. 2013]. A aplicação de clientes de sincronização desenvolvidos para usuários domésticos não atende às necessidades de ambientes organizacionais. As principais barreiras arquiteturais encontradas são: (i) ausência de integração com sistemas de armazenamento privado; (ii) necessidade de cópia local de todos os arquivos compartilhados; e (iii) latência de sincronização entre arquivos da nuvem e repositórios locais.

3. Arquitetura CHSAN

A arquitetura do Cliente Híbrido para Sincronização de Arquivos em Nuvem (CHSAN), resumida na Figura 2, está alinhada com os principais clientes de sincronização de arquivos na nuvem disponíveis à comunidade. Os módulos comuns em suas arquiteturas (armazenamento, transporte, observador e repositório de metadados) [Houston and Ferdowsi 2014] [Besen et al. 2015] foram estendidos por CHSAN.

No módulo de armazenamento, incorporou-se um sistema de arquivos virtual e *cache* seletiva, enquanto o módulo de transporte foi adaptado para encaminhar dados a um armazenamento externo legado (além dos repositórios da nuvem). O sistema de arquivos virtual permite que o cliente tenha acesso à totalidade dos índices dos arquivos armazenados na nuvem, mesmo sem nunca tê-los sincronizado em sua integralidade. Assim, embora o volume de armazenamento disponível seja inferior ao total necessário, o cliente ainda tem conhecimento sobre a existência de todos os arquivos e informações referentes as alterações realizadas. Esse ponto diferencia CHSAN dos clientes tradicionais que informam a existência de arquivos somente quando sincronizados com o repositório principal. A *cache* seletiva agiliza o acesso aos dados mais usados pelo cliente (atuando no requisito de latência de acesso). Dispositivos legados, como *Network Attached Storage* (NAS), recebem informações do módulo de transporte, como opção para minimização do uso do enlace à Internet e alternativa para comunicação em enlaces dedicados.

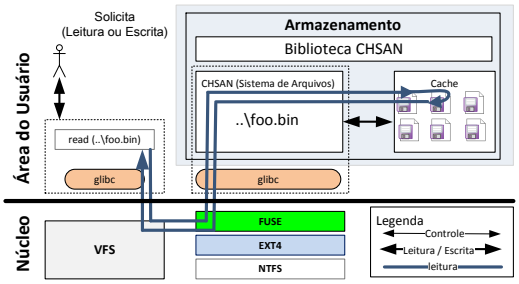


Figura 3. Visão detalhada do módulo de armazenamento CHSAN.

Para aplicativos finais, a existência de CHSAN é transparente (aplicativos utilizam a interface padrão do sistema operacional). Os módulos e o fluxo de dados são individualmente discutidos na sequência.

3.1. Armazenamento

O módulo de armazenamento do CHSAN é responsável pela apresentação do sistema de arquivos bem como pelas operações de leitura e gravação, combinando sistema de arquivos virtuais (VFS – *Virtual File System*) com FUSE (*Filesystem in Userspace*) para criar um sistema de arquivos customizado em espaço de usuário, sem a necessidade de intervenção no núcleo. O módulo é constituído de: (i) espaço de armazenamento local; *cache*, alimentada de forma seletiva; (ii) biblioteca de acesso, que disponibiliza funções para consulta, escrita e leitura de arquivos; e (iii) sistema de arquivos CHSAN, que exporta para o sistema operacional uma estrutura de diretórios e arquivos como uma unidade de disco local. Tal exportação é análoga, do ponto de vista de funcionalidade e compatibilidade, à estrutura de diretórios dos clientes de sincronização de arquivos em nuvem.

Na Figura 3 é representado o fluxo de uma operação submetida por um aplicativo usuário. Na parte inferior da figura, modo núcleo, têm-se os módulos VFS e FUSE e, na parte superior, área do usuário, o aplicativo do cliente, a biblioteca *glibc* e o CHSAN. O fluxo é iniciado pelo aplicativo do cliente que gera a operação em um arquivo. Esta solicitação, ainda na área do usuário, utiliza da biblioteca padrão do sistema para acesso a arquivos locais, neste caso, *glibc*. Através desta biblioteca, a solicitação é encaminhada ao modo núcleo processada pelos módulos VFS e FUSE, assim como acontece com operações de outros sistemas de arquivos (*e.g.*, EXT4 e NTFS). O módulo de armazenamento CHSAN, quando acionado, interage com o repositório de metadados por meio desta biblioteca e identifica a localização do arquivo solicitado, caso ele exista, ou provoca a criação de um novo índice no repositório de metadados. O fluxo de retorno do conteúdo somente é iniciado se o arquivo estiver na *cache* local. Caso contrário, o módulo de armazenamento aguarda até que o conteúdo seja copiado localmente. A transferência do conteúdo solicitado à *cache* local é responsabilidade do módulo de transporte auxiliado pelo módulo observador.

3.2. Módulos Observador e Transporte

O módulo observador tem como objetivo a execução regular de tarefas assíncronas para atualizar a base local de metadados frente às alterações detectadas nos repositórios de armazenamento da nuvem. Estas atualizações são realizadas nos índices do sistema de arquivos do CHSAN, invalidando, quando necessário, as entradas na *cache*. Além disso, o módulo observador solicita o *upload* para o armazenamento externo (NAS ou repositório da nuvem)

quando existe alguma entrada pendente no repositório de metadados. É importante ressaltar que CHSAN não foi concebido para ser utilizado sem o armazenamento em nuvem, uma vez que este é base para a composição e manutenção dos metadados.

O módulo de transporte do CHSAN diferencia-se das arquiteturas tradicionais de sincronização, pois, uma vez parametrizado, pode interagir com diferentes repositórios externos (provedores de nuvem ou repositórios legados). Cabe a ele a identificação do repositório com menor latência e a transferência dos arquivos remotos para a *cache* local. Assim, o usuário pode estar conectado na rede privada da organização e recuperar o arquivo de um repositório privado, ou estar conectado à Internet e recuperar o arquivo diretamente da nuvem. Desta forma, o acesso aos arquivos compartilhados maximiza o tráfego de dados e não está limitado à capacidade de armazenamento do dispositivo local ou à restrições de seleções manuais de sincronização.

3.3. Repositório de Metadados

O repositório de metadados, gerido pelo módulo observador, é o responsável pela indexação da totalidade dos arquivos compartilhados na nuvem. Cada arquivo possui somente um índice no repositório de metadados, mas pode estar replicado em *cache* local, sistema de armazenamento privado e na nuvem. Os metadados são concentrados em um banco de dados, contendo informações como identificadores, caminho do objeto (hierarquia), tipo do objeto (arquivo ou diretório), tamanho desse objeto, as datas de criação e alteração, bem como um controle de operações realizadas e pendentes. De acordo com a situação atual da informação sobre os objetos (sincronizado com a nuvem, criado ou pendente de sincronização), o módulo observador aciona a sincronização. Em suma, CHSAN baseia-se nos metadados para montar o sistema de arquivos virtual disponibilizado aos usuários, permitindo que o mesmo visualize a totalidade do sistema de arquivos compartilhado, independente da localização física deste arquivo no momento da requisição.

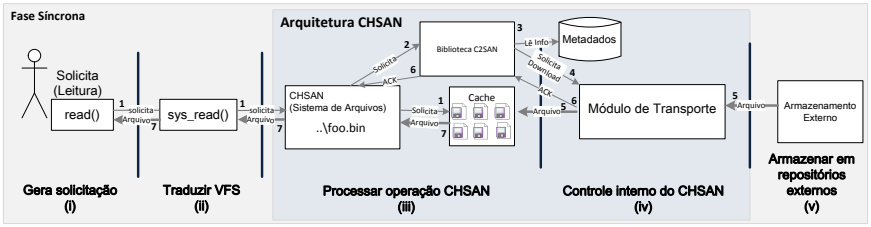
3.4. Operações de Leitura e Escrita

As operações de leitura e escrita são as demandas mais complexas gerenciadas por CHSAN, envolvendo todos os módulos da arquitetura. O fluxo de execução é decomposto em uma fase síncrona e outra assíncrona. Na fase síncrona, o aplicativo é bloqueado até que o CHSAN processe a operação. A fase assíncrona permite que o aplicativo continue executando enquanto o CHSAN processa as operações pendentes.

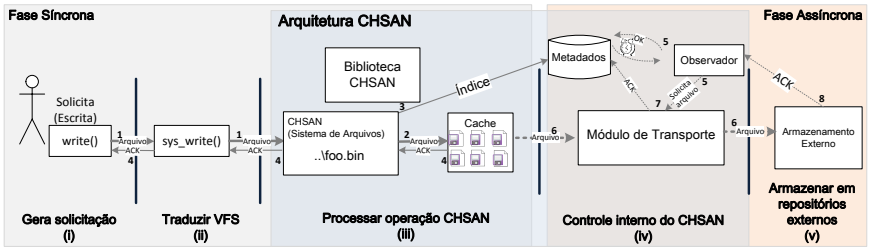
Embora as operações de leitura e escrita sejam tratadas de formas diferentes, elas seguem um fluxo similar de execução, dividido em cinco etapas: (i) gerar solicitação; (ii) traduzir VFS; (iii) processar operação CHSAN; (iv) controle interno do CHSAN; (v) armazenar em repositórios externos. A etapa (i) é executada na área do usuário e iniciada pela aplicação. No modo núcleo, a etapa (ii) recupera as informações da solicitação advindas da aplicação do usuário e traduz para VFS. No CHSAN, novamente no modo usuário, a etapa (iii) representa a operação do módulo de armazenamento da arquitetura. A etapa (iv) é de responsabilidade dos módulos observador e transporte, enquanto a etapa (v) é caracterizada pela sincronização dos dados entre a *cache* e os repositórios.

Na Figura 4 ilustra-se o fluxo de processamento das operações de leitura e escrita (representando em passos). As etapas (i)-(v) são representados pelo fluxo de execução (da esquerda para a direita). Na Figura 4(a) tem-se o fluxo de uma operação de leitura, em que um aplicativo gera a solicitação e a envia (identificada no passo 1). Se o arquivo solicitado

já existe na *cache* local e está coerente com a informação existente no índice de arquivos (tamanho e data de escrita são consultados no conjunto de metadados), ele é imediatamente retornado (passo 7) à aplicação. Neste caso, a resposta ao usuário é possível com o processamento das etapas (i) e (ii) simplesmente. Se o arquivo solicitado não estiver na *cache*, a etapa (iii) é iniciada. O sistema de arquivos do CHSAN solicita ao módulo de transporte o *download* do arquivo do repositório externo (servidor NAS ou repositório da nuvem). No módulo de transporte, os passos 2 (solicitação do arquivo à biblioteca CHSAN), passo 3 (leitura dos metadados) e passo 4 (efetua *download*) são processados. O fluxo segue na etapa (iv), desta vez, sob responsabilidade do módulo de transporte. A leitura do armazenamento NAS é prioritária em relação ao repositório na nuvem, portanto, se o arquivo estiver atualizado no NAS, os passos 5, 6 e 7 são executados e o arquivo entregue à aplicação. Finalmente, se o arquivo só estiver na nuvem, o módulo de transporte atualiza o arquivo no armazenamento NAS (passo 5), caracterizando a etapa (v), e o fluxo de retorno do arquivo é concluído (passos 6 e 7).



(a) Fluxo para leitura de um arquivo usando CHSAN.



(b) Fluxo para escrita de um arquivo usando CHSAN.

Figura 4. Exemplos de fluxo de leitura e escrita usando CHSAN.

Enquanto o processo de leitura acontece de forma síncrona, o processo de escrita (Figura 4(b)) possui uma fase assíncrona. Uma escrita usando CHSAN ocorre inicialmente na *cache* da arquitetura. Os blocos de dados são redirecionados (passo 1) à *cache* local, passando pelo núcleo. O índice do arquivo é atualizado na base de metadados e uma confirmação é enviada (passo 4) ao sistema operacional, finalizando a operação. A fase assíncrona (passo 5) é executada pelo observador em intervalos regulares para verificação de pendências de sincronização na base de metadados. Quando necessário, o módulo de transporte é acionado para realizar o *upload* aos repositórios de armazenamento externo (NAS e provedor de nuvem). Ao final do passo 6, os armazenamentos externos e o metadado local é atualizado.

4. Análise Experimental

Como prova de conceito, um protótipo da arquitetura foi implementado e analisado comparando métricas com clientes de sincronização populares. Os experimentos focaram na quantificação da sobrecarga ocasionada por CHSAN quando comparado com clientes populares de sincronização e tecnologias para repositório distribuídos.

4.1. Ambiente de Testes

O ambiente de testes foi composto por máquinas virtuais, hospedadas em dois equipamentos HP EliteBook 840 com Core $I5-4300U$ de 2.5 GHz, 16 GB RAM, disco local de 500 GB SATA, executando o sistema operacional Windows 7 e software de virtualização VMware player $7.1.2$. Cada máquina virtual foi configurada com 2 vCPUs, 8 GB RAM e 80 GB para armazenamento. Como sistema de arquivos, foi utilizado o NTFS com configurações padrão. Os equipamentos foram interconectados em uma LAN com *switch* Gigabit, e uma conexão Wifi foi configurada com um roteador D-LINK DWR-922 ($144,4$ Mbps).

Os testes de escrita no armazenamento USB foram realizados em um dispositivo móvel Kingston Data Travel 101 de 64 Gb, acessível na máquina virtual. Testes com escritas remotas foram realizados em repositório compartilhado, hospedado em um equipamento com a mesma descrição, executando Windows 10. No protótipo de CHSAN, utilizou-se o banco de dados Microsoft SQLServer 2014 Express para armazenamento dos metadados.

4.2. Métricas e Comparações

As métricas selecionadas para discussão caracterizam o tempo necessário para executar operações e a vazão média do sistema. O tempo de execução é relacionado com a expectativa de desempenho, sendo crucial para sincronização entre usuários colaborativos, enquanto a vazão média do sistema quantifica a capacidade máxima atingida para cada configuração estudada. O presente artigo segue a metodologia de trabalhos anteriores, executando operações de leitura e escrita com diferentes tamanhos de arquivos [Kanaujia and Giridhar 2012] (1 MB, 50 MB, 100 MB e 200 MB). Para geração e submissão de arquivos, foi utilizada a ferramenta *postmark* [Katcher 1997] e um programa para geração de arquivos aleatórios. Para cada experimento, foram realizadas dez rodadas, e os gráficos têm representados os valores médios, o desvio padrão, valores máximos e mínimos (intervalo de confiança de 95%). Além de CHSAN, os testes foram promovidos com seis conjuntos de sistema de arquivos:

- *Nativo local*: leitura e escrita realizadas no mesmo dispositivo de armazenamento (com NTFS) que o sistema operacional está localizado. Este cenário é utilizado como linha de base para análise de desempenho (identificado por *Local* nos gráficos).
- *Nativo LAN cabeada*: operações realizadas em um repositório remoto com sistema de arquivo CIFS (sobre NTFS), conectado por um *switch* Gigabit (legenda *LAN cab*).
- *Nativo LAN Wifi* operações realizadas em um repositório (CIFS sobre NTFS) acessado remotamente através de um *access point* doméstico, configurado em modo infraestrutura com conectividade máxima de $144,4$ Mbps (legenda *LAN Wifi*).
- *Nativo USB*: escrita e leitura realizadas em um dispositivo móvel de armazenamento conectado por uma interface USB 2.0 com sistema de arquivos FAT32 (legenda *USB*).
- *Dokan C++* [5]: sistema de arquivo FUSE em C++ que dispõe de um volume remoto como um diretório do sistema de arquivos local.
- *Dokan .Net*: possui funcionalidades iguais ao cenário anterior. Entretanto, foi implementado com a linguagem C# .NET, mesma utilizada pelo protótipo CHSAN.

[5]Dokan, disponível em `https://github.com/dokan-dev/`.

4.3. Discussão dos Resultados

A Figura 5 e a Tabela 1 contém os valores obtidos com a execução do *benchmark postmark* para escrita, leitura e deleção de 10 arquivos (por rodada) com conteúdo aleatório. O primeiro conjunto de testes visa avaliar o desempenho das operações sem atualização imediata do repositório da nuvem. Os cenários *Nativo local*, *Nativo USB*, *Nativo LAN cabeada* e *Nativo LAN Wifi* não alteram seu modo de funcionamento pois os dois primeiros têm como objetivo o armazenamento local e, os dois últimos, não são sincronizados com repositórios na nuvem. Os cenários *Dokan C++* e *Dokan .Net* foram parametrizados para não sincronizarem seus arquivos com o repositório na nuvem. Em CHSAN, as operações realizadas são atendidas pelas etapas (i), (ii) e (iii) (Figura 4). Nos cenários com sistema de arquivos virtual (*Dokan C++*, *Dokan .Net* e CHSAN), o impacto no desempenho é oriundo das operações para acesso ao repositório de metadados. Os cenários com sistemas nativos não precisam deste repositório e, naturalmente, apresentam tempos de acesso inferiores aos demais.

O tempo de execução das operações (em segundos) nos diferentes repositórios é resumido na Figura 5. Os cenários *Nativo LAN cabeada* e *Nativo LAN Wifi* representam a linha de base de acesso a sistemas de armazenamento privado implantados em uma LAN. Embora possua um mecanismo com maior número de operações (discutido na Seção 3), a sobrecarga imposta por CHSAN, quando comparado ao cenário *Nativo LAN cabeada*, é somente perceptível para operações com arquivos pequenos (1 MB), sendo suavizada para arquivos maiores (aproximadamente 2.9, 3.7 e 3.9 maior para cenários com 50 MB, 100 MB e 200 MB). Ainda, o tamanho dos arquivos não influencia diretamente o tempo das operações.

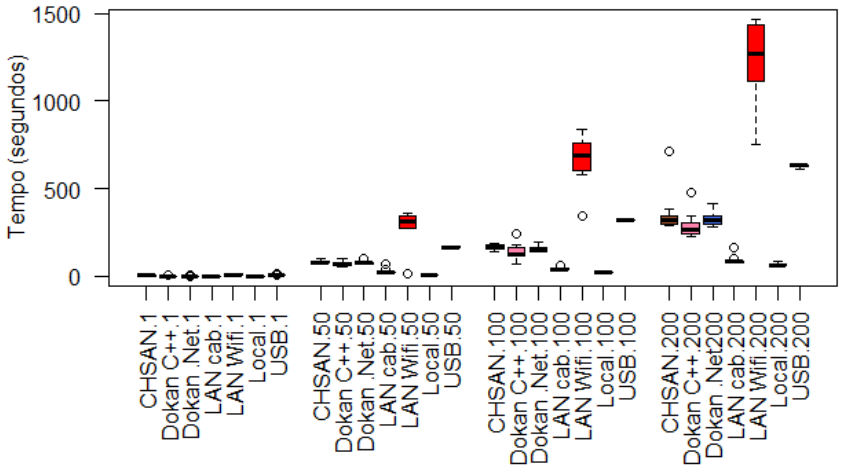


Figura 5. Tempo de execução (em segundos) dos sistemas analisados.

Quando comparado aos cenários com escrita direta no sistema de arquivos virtual, que possuem repositório de metadados, identificados pelas legendas *Dokan C++* e *Dokan .Net*, CHSAN obteve resultados competitivos. *Dokan .Net* apresentou uma sobrecarga aproximada de 36%, 10%, 5%, 8% e 6%, para operações com arquivos de 1 MB até 200 MB, respectivamente. Esses valores representam o tempo necessário para atualização do repositório de metadados. CHSAN obteve um menor tempo médio de execução das operações quando comparado aos cenários *Nativo Wifi* e *Nativo USB*. Em suma, a sobrecarga acrescentada por CHSAN quando comparada às linhas de base (*Nativo local* e *Nativo LAN cabeada*) é esperada, devido às operações necessárias para controle e gerenciamento dos metadados.

Entretanto, quando comparado a cenários logicamente próximos (*Dokan C++* e *Dokan .Net*), CHSAN obteve um desempenho aceitável. Quando comparado aos cenários *Nativo USB* e *Nativo LAN Wifi*, CHSAN obteve-se um desempenho superior em todos os casos.

	1 MB		50 MB		100 MB		200 MB	
Cenário	$\varnothing \pm \sigma$	min; max	$\varnothing \pm \sigma$	min; max	$\varnothing \pm \sigma$	min; max	$\varnothing \pm \sigma$	min; max
Nativo local	10 ± 0	10; 10	71,43 ± 9,09	62,50; 83,33	49,75 ± 10,09	37,04; 67,02	31,49 ± 3,85	23,53; 35,71
Nativo LAN cabeada	8,33 ± 2,72	3,33; 10	20,26 ± 5,77	7,35; 25	23,54 ± 3,38	15,63; 27,03	23,29 ± 4,33	12,20; 26,67
Nativo LAN Wifi	1,02 ± 0,37	0,5; 1,25	6,44 ± 15,31	1,39; 50	1,59 ± 0,49	1,19; 2,88	1,68 ± 0,40	1,37; 2,66
Nativo USB	1,25 ± 0,21	0,83; 1,67	3,01 ± 0,09	2,89; 3,16	3,14 ± 0,03	3,09; 3,17	3,18 ± 0,04	3,15; 3,27
Dokan C++	7,75 ± 2,99	2,50; 10	7,18 ± 1,29	5,10; 8,77	7,90 ± 2,99	4,20; 15,38	7,29 ± 1,46	4,21; 8,93
Dokan .Net	5,83 ± 2,26	3,33; 10	6,75 ± 0,73	5,05; 7,69	6,53 ± 0,69	5,18; 7,35	6,14 ± 0,81	4,85; 7,19
CHSAN	3,39 ± 0,87	3,30; 5	6,07 ± 0,64	4,95; 6,76	6,19 ± 0,54	5,43; 6,99	5,92 ± 1,21	2,81; 6,87

Tabela 1. Vazão alcançada pelos sistemas analisados (em KB/s).

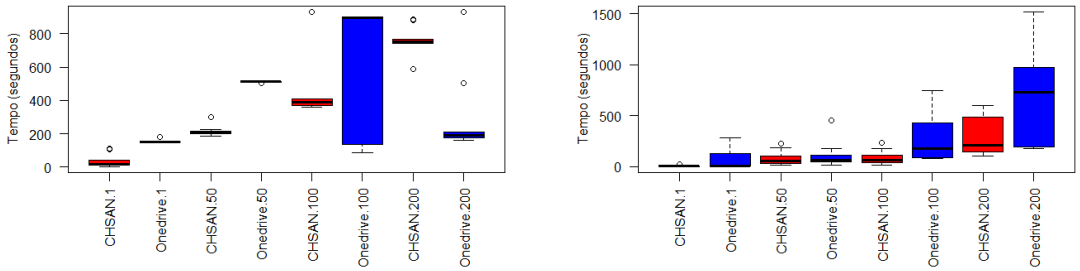
Na Tabela 1, estão representados os valores para a vazão (kilobytes por segundo) alcançada durante a execução dos testes nos cenários propostos. O comportamento dos resultados de CHSAN perante os cenários *Nativo local* e *Nativo LAN cabeada* é semelhante ao discutido na Figura 5. Entretanto, a vazão média alcançada por CHSAN é comparável aos sistemas de escrita nativa sobre sistemas de armazenamento virtuais. É importante ressaltar que, independente do volume de dados escritos e lidos no sistema, CHSAN manteve uma vazão superior aos sistemas *Nativo USB* e *Nativo LAN Wifi*. Ainda, a vazão obtida por CHSAN é de aproximadamente $6{,}18$ KB/s para arquivos entre 50 e 200 MB, com valores menores para arquivos de 1 MB ($3,9$ KB/s).

Os resultados do primeiro conjunto de testes comprovam a eficiência do CHSAN nas operações de escrita, leitura e deleção de arquivos sem atualização imediata do repositório na nuvem. Arquiteturas dos clientes de sincronização de arquivos na nuvem baseiam-se em sistemas de arquivos *Nativo local*, otimizando seus resultados neste contexto, porém, ignoram o conteúdo dinâmico do repositório de metadados e, consequentemente, suas vantagens.

O segundo conjunto de testes (Figura 6) avaliou o desempenho das operações em arquivos compartilhados localizados unicamente no sistema de armazenamento na nuvem. Em CHSAN, o acesso a estes arquivos só é possível com a aplicação das cinco etapas descritas na Figura 4. Embora algumas operações possam ser realizadas de forma assíncrona, o método aplicado para medição do tempo assegurou-se que o arquivo esteja todo na nuvem. Considerando a média, o CHSAN superou o cliente para sincronização de arquivos *OneDrive* para arquivos de 1 MB, 50 MB e 100 MB, nas operações escrita, Figura 6(a), e leitura, Figura 6(b). A exceção foi o resultado da escrita de arquivos com 200 MB: *OneDrive* obteve tempo inferior ao CHSAN, entretanto, o desvio padrão indica uma maior dispersão nos resultados, fato indesejável quando analisado o requisito de latência de acesso.

4.4. Discussão e Parametrização da Arquitetura

O CHSAN ocupa uma área de armazenamento de *cache* configurável que independe do tamanho dos dados armazenados no repositório da nuvem. No estudo de caracterização de armazenamento em nuvem realizado por [Gonçalves et al. 2014] foi identificado que 333



(a) Operação de escrita de um arquivo na nuvem. (b) Operação de leitura de um arquivo na nuvem.

Figura 6. Operações em arquivos localizados na nuvem (CHSAN vs. *OneDrive*).

usuários possuíam 3 milhões de arquivos totalizando aproximadamente $1,38$ TB de dados. No protótipo de CHSAN, usando o banco de dados Microsoft SQLServer 2014, esse volume seria indexado e disponibilizado, consumindo apenas $2,2$ GB de área de armazenamento.

O fluxo total de entrada e saída para o armazenamento externo é diminuído, pois somente são trafegados arquivos solicitados, abordagem distinta da realizada pelos clientes de sincronização convencionais que mantém a sincronização independente do uso. Além disso, a implementação do CHSAN com um armazenamento NAS reduz o tráfego à Internet para a leitura de arquivos por múltiplos usuários colaborativos. Os clientes de um mesmo compartilhamento NAS deixam de buscar os dados da nuvem e passam a atualizar a *cache* com base nas versões locais. Entretanto, o contato com a nuvem é necessário a cada escrita para disponibilizar aos usuários externos as informações sobre o procedimento realizado. A equação $v = b_e * \alpha(1+n)$ pode ser utilizada para estimar o volume de dados trafegados com os repositórios de armazenamento na nuvem, sendo v o volume total de dados trafegados, b_e o volume de bytes escritos no provedor de nuvem, n o número de clientes a sincronizar, e $\alpha \leq 1$ o coeficiente de compressão. Na arquitetura híbrida de CHSAN, v passa a ser guiado por $v = \alpha * b_e$, sendo independente do número de clientes colaborativos.

5. Trabalhos Relacionados

A literatura especializada compreende trabalhos sobre caracterização de uso de armazenamento na nuvem e técnicas que melhoram o desempenho de clientes e protocolos.

Quanto à caracterização, Drago [Drago et al. 2012] e [Drago et al. 2013] identificaram o tráfego de dados da ferramenta *DropBox* em um ambiente universitário, indicando que o uso de recursos de rede para sincronização de arquivos alcançou um terço do tráfego para visualização de vídeos no cenário estudado. Estes trabalhos caracterizaram o comportamento dos usuários contabilizando o volume de dados, tamanho médio dos arquivos e atividades de sincronização. Eles também forneceram uma perspectiva sobre o uso de clientes de sincronização em usuários domésticos, embasando a realização da presente proposta, porém, não se preocuparam com ambientes organizacionais com múltiplos usuários, discutido na Seção 2. Nestes ambientes, a flexibilidade e a heterogeneidade são características essenciais para um sistema de arquivos [Tate et al. 2005] [Shepler et al. 2003] [Hertel 2003] [French and Team 2007], enquanto a escalabilidade, a alta disponibilidade e a segurança são desejáveis [Tate et al. 2005]. CHSAN atende às características essenciais de sistemas de arquivos e ainda se preocupa com a escalabilidade e a alta disponibilidade, combinando três aspectos: o acesso ao disco local, aos sistemas de armazenamento privado, e ao armazenamento na nuvem.

Especificamente sobre armazenamento na nuvem, algumas soluções foram desenvolvidas com foco na escalabilidade dos sistemas de armazenamento GFS [Ghemawat et al. 2003], S3FS [Muniswamy-Reddy et al. 2010] e Hadoop [Shvachko et al. 2010]. Assim como CHSAN, essas soluções foram construídas com base em repositórios de metadados e sistemas de arquivos distribuídos. Por sua vez, CIFS [Hertel 2003], NFS [Shepler et al. 2003] e SAN [Tate et al. 2005] foram concebidos para atender ambientes organizacionais de menor escala. Em função da proposta para escala global, os sistemas de arquivos baseados em metadados e acesso a objetos, discutidos anteriormente, formaram a base do armazenamento em repositórios de nuvem.

Quanto às propostas para otimização do desempenho de clientes de sincronização, [Wang et al. 2012] e [Li et al. 2013] apresentaram técnicas para diminuir o tráfego total, tempo de acesso aos arquivos e *overuse*, no qual o tráfego de dados é maior que a quantidade de dados efetivos. O cliente de sincronização *Dropbox* destaca-se neste quesito, pois é o único que fornece a possibilidade de sincronização direta entre clientes localizados em uma mesma rede lógica por meio do protocolo *LanSync*. Nessa linha, CHSAN inova ao permitir a sincronização entre clientes concentrados em redes privadas, estando na mesma rede lógica ou não.

6. Conclusão

A adoção de serviços de armazenamento em nuvem por usuários domésticos e organizações é uma realidade, por oferecer maior disponibilidade aos dados, segurança e colaboração para os usuários. Entretanto, os clientes de sincronização de arquivos populares não foram desenvolvidos para atender os requisitos específicos de organizações com múltiplos usuários colaborativos, geograficamente distribuídos e concentrados em redes privadas, quando a latência no acesso aos arquivos é um ponto crítico para a sincronização. Nesse contexto, o presente artigo apresentou a arquitetura de um Cliente Híbrido para Sincronização de Arquivos em Nuvem – CHSAN – que combina o uso do disco local, de sistemas de armazenamento privado e do armazenamento na nuvem. CHSAN estendeu as características básicas dos clientes de sincronização de arquivos na nuvem, fornecendo soluções para as três principais limitações de uso destes clientes nas organizações: (i) ausência de integração com sistemas de armazenamento privado; (ii) necessidade de cópia local de todos os arquivos compartilhados; e (iii) latência de sincronização entre arquivos da nuvem e repositórios locais.

A análise experimental concentrou-se em dois cenários, (i) avaliação do impacto no desempenho de operações sobre arquivos no acesso ao repositório de metadados e (ii) avaliação do desempenho das operações em arquivos compartilhados, localizados unicamente na nuvem. Por possuir um mecanismo específico para controle de metadados, CHSAN apresentou uma sobrecarga nas operações com arquivos de 1 MB quando comparado a sistemas de acesso local. Nos demais testes, CHSAN obteve resultados motivadores, chegando a ser 36% melhor do que *Dokan .Net*. A taxa de vazão atingiu $6,18$ KB/s para arquivos entre 50 e 200 MB. Finalmente, CHSAN foi melhor que *OneDrive* tanto na escrita quanto na leitura, para arquivos de 1 MB, 50 MB e 100 MB. Como trabalhos futuros, destacam-se um estudo específico sobre a adequação do método de alimentação da *cache* seletiva automática, e a comparação do CHSAN com outros clientes populares (*Google Drive* e *Dropbox*).

Referências

Besen, A., Cheong, H., Mueller, H., Pape, F., and Wurtz, D. (2015). Sharing and synchronizing electronically stored files. US Patent 8,949,179.

Douceur, J. R. and Bolosky, W. J. (1999). A large-scale study of file-system contents. In *Proc. of the ACM SIGMETRICS*, pages 59–70, New York, NY, USA. ACM.

Drago, I., Bocchi, E., Mellia, M., Slatman, H., and Pras, A. (2013). Benchmarking personal cloud storage. In *Proc. of the Internet Measurement Conference*, IMC '13, pages 205–212. ACM.

Drago, I., Mellia, M., M. Munafo, M., Sperotto, A., Sadre, R., and Pras, A. (2012). Inside dropbox: Understanding personal cloud storage services. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, pages 481–494, New York, NY, USA. ACM.

French, S. M. and Team, S. (2007). A new network file system is born: Comparison of SMB2, CIFS and NFS. In *Linux Symposium*, page 131, Ottawa, Canada.

Ghemawat, S., Gobioff, H., and Leung, S.-T. (2003). The google file system. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 29–43. ACM.

Gonçalves, G., Drago, I., Couto da Silva, A. P., Marques de Almeida, J., and Borges Vieira, A. (2014). Caracterização e modelagem da carga de trabalho do dropbox. *Anais do 32º SBRC*, pages 545–558.

Hertel, C. (2003). *Implementing CIFS: The Common Internet File System*. Prentice Hall Professional.

Houston, D. and Ferdowsi, A. (2014). Network folder synchronization. US Patent 8,825,597.

Kanaujia, V. and Giridhar, C. (2012). FUSE: Python for development of storage efficient filesystem.

Katcher, J. (1997). Postmark: A new file system benchmark. Technical report, Network Appliance, TR3022.

Katzer, M. and Crawford, D. (2013). Office 365: Moving to the cloud. In *Office 365*, pages 1–23. Springer.

Li, Z., Wilson, C., Jiang, Z., Liu, Y., Zhao, B., Jin, C., Zhang, Z.-L., and Dai, Y. (2013). Efficient batched synchronization in dropbox-like cloud storage services. In Eyers, D. and Schwan, K., editors, *Middleware 2013*, volume 8275 of *Lecture Notes in Computer Science*, pages 307–327. Springer Berlin Heidelberg.

Mell, P. M. and Grance, T. (2011). Sp 800-145. the NIST definition of cloud computing. Technical report, NIST, Gaithersburg, MD, United States.

Muniswamy-Reddy, K.-K., Macko, P., and Seltzer, M. (2010). Provenance for the cloud. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, FAST'10, pages 15–14.

Naldi, M. and Mastroeni, L. (2013). Cloud storage pricing: A comparison of current practices. In *Proc. of the International Workshop on Hot Topics in Cloud Services*, HotTopiCS '13, pages 27–34. ACM.

Shepler, S., Eisler, M., Robinson, D., Callaghan, B., Thurlow, R., Noveck, D., and Beame, C. (2003). Network file system (nfs) version 4 protocol. *Network*.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE.

Smith, D. M., Petri, G., Natis, Y. V., Scott, D., Warrilow, M., Heiser, J., Bona, A., Toombs, D., Yeates, M., Bova, T., and Lheureux, B. J. (2013). Cloud computing affects all aspects of it. Technical report, Gartner Group.

Tate, J., Lucchese, F., Moore, R., and TotalStorage, S. (2005). *Introduction to storage area networks*. IBM Corporation, International Technical Support Organization.

Wang, H., Shea, R., Wang, F., and Liu, J. (2012). On the impact of virtualization on dropbox-like cloud file storage/synchronization services. In *Proc. of the IEEE 20th International Workshop on Quality of Service*, pages 11:1–11:9, Piscataway, NJ, USA.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18.

Um SLA 3D para o Escalonamento Multiusuário Sobre Múltiplos Provedores de IaaS

Cristiano C. A. Vieira[1], **Luiz F. Bittencourt**[1], **Edmundo R. M. Madeira**[1]

[1]Universidade de Campinas - Instituto de Computação
Av. Albert Einstein, 1251 - Cidade Universitária, Campinas/SP - Brasil

cargemon@facom.ufms.br, bit@ic.unicamp.br, edmundo@ic.unicamp.br

Abstract. *Customers of Cloud providers run applications with different Quality of Service (QoS) requirements. However, the main item treated in SLAs is currently the time of use and the implications for their violation, neglecting other important factors in an SLA. To address this issue, in this paper, we introduce an SLA that considers three important aspects of QoS: reliability, performance and security. The goal is to schedule virtual machine requests from users to public IaaS providers in order to decrease the monetary cost without violating the requests QoS. We present an integer linear program (ILP) formulation and a heuristic to compute the scheduling. In addition, we discuss the experimental results obtained from the implementation of our strategy.*

Resumo. *Usuários de provedores de nuvens executam aplicações com diferentes necessidades de QoS. Contudo, o principal item tratado nos SLAs atualmente é o tempo de utilização e as implicações quanto a sua violação, negligenciando outros fatores importantes em um SLA. Para tratar esta questão, neste trabalho, introduzimos um SLA que considera três importantes aspectos de QoS: confiabilidade, desempenho e segurança. O objetivo é escalonar requisições de máquinas virtuais de múltiplos usuários em provedores públicos de IaaS obtendo o menor custo monetário sem violar a QoS das requisições. Apresentamos uma PLI e uma heurística para computar o escalonamento. Além disso, apresentamos e avaliamos os resultados experimentais obtidos a partir da implementação da nossa estratégia.*

1. Introdução

A computação em nuvem atrai cada vez mais usuários a consumirem e provedores a oferecerem serviços em nuvem por meio do modelo *pay-as-you-go*. Os usuários terceirizam suas demandas computacionais para os provedores públicos e pagam de acordo com sua utilização. Neste cenário, um crescente número de empresas está adotando serviços de computação em nuvem, tais como infraestrutura (IaaS), plataforma (PaaS) e software (SaaS) [Zhang et al. 2010]. Considerando serviços de infraestrutura, usuários alugam capacidade de processamento a partir de provedores de IaaS (Amazon EC2[1], FlaxiScale [2], Google Compute Engine[3], ElasticHosts[4], CloudSigma[5] e JoyentCloud[6]) sob a

[1]Elastic Cloud Computing - hwwps://aws.amazon.com/ec2
[2]FlexiScale Cloud Comp and Hosting - http://www.flaxiscale.com
[3]Google Compute Engine - https://cloud.google.com/compute/
[4]ElasticHosts - http://www.elastichosts.com/
[5]CloudSigma - https://www.cloudsigma.com/
[6]JoyentCloud - https://www.joyent.com/

forma de máquina virtual (VM - *Virtual Machine*) onde podem implantar suas aplicações sob diferentes modelos de cobrança, tais como OD (*On Demand*), RE (*Reserved*) e SP (*Spot*) [Vieira et al. 2015]. Instâncias sob o modelo OD possuem o maior custo. Contudo, o maior nível de confiabilidade. São interrompidas apenas pelo usuário. Instâncias RE possuem o custo menor comparado ao do OD. Contudo, o usuário precisa pagar uma taxa para reservar sua utilização por um período de tempo. Instâncias SP possuem o menor custo e também a menor confiabilidade, pois podem ser interrompidas pelos provedores caso o valor oferecido pelo usuário seja inferior ao valor cobrado pelo provedor.

Existem inúmeros objetivos complexos intrínsecos aos pontos de vista do usuário e do provedor na comercialização através do modelo *pay-as-you-go*. Sob o ponto de vista do usuário, o objetivo é beneficiar-se da computação em nuvem obtendo alta disponibilidade, redução de custos, balanceamento de carga e melhoria na tolerância a falhas. A seleção dos recursos depende das necessidades dos usuários e de suas aplicações.

Várias características em um SLA (*Service Level Agreement*) oferecido pelo provedor de IaaS devem ser levadas em consideração durante o escalonamento das VMs para o usuário executar suas aplicações, tais como: confiabilidade, desempenho, segurança, localização, tempo de execução e outras. Ao consideramos diferentes itens no SLA o usuário consumidor da nuvem encontra dificuldades em escolher o melhor conjunto de recursos para atender suas necessidades e reduzir os custos. Neste contexto, várias plataformas, como os *Brokers*, têm sido propostas para oferecer o serviço de descoberta e a alocação de recursos sobre múltiplos provedores de IaaS ajudando os usuários neste processo de decisão. Contudo, nenhuma considera o escalonamento de múltiplos usuários e diferentes características no SLA. Vieira et. al [Vieira et al. 2015] propuseram uma PaaS que oferece o serviço de escalonamento considerando duas características no SLA: confiabilidade e desempenho, e realiza o escalonamento de requisições de apenas um usuário.

Ao considerarmos vários usuários durante o escalonamento, a questão de segurança é evidente. Diferentes usuários requerem diferentes políticas de segurança para executar suas aplicações. Em geral, alcançar um nível satisfatório de segurança para serviços fornecidos em nuvem é um desafio dada a complexidade, heterogeneidade e natureza dinâmica das necessidades dos usuários e disponibilidade dos serviços oferecidos pelos provedores [Silva and de Geus 2015]. Neste trabalho, abordamos o problema de escalonamento de requisições de VMs de usuários distintos sobre recursos de múltiplos provedores de IaaS. Nossa contribuição é a extensão da PaaS apresentada em [Vieira et al. 2015] por tratar requisições de vários usuários e considerar a segurança no escalonamento. Para isso propomos neste trabalho: a) Um SLA com três parâmetros: confiabilidade, desempenho e segurança; b) Uma PLI para computar o escalonamento utilizando o SLA proposto; e c) Um algoritmo heurístico para evitar violação de QoS; e d) Avaliação dos resultados experimentais.

O restante do texto está organizado como a seguir: na próxima seção, descrevemos uma visão geral sobre os trabalhos relacionados ao tema. Em seguida, na Seção III, caracterizamos as necessidades dos usuários de provedores de IaaS e definimos o problema abordado. Na Seção IV, apresentamos nossa proposta de escalonamento multiusuário. Na Seção V, apresentamos os resultados. Na Seção VI apresentamos nossas conclusões.

2. Trabalhos Relacionados

Várias plataformas como *Brokers* foram propostas para otimizar a seleção de recursos e escalonar as requisições dos usuários sobre os provedores de IaaS. Em [Nair et al. 2010] são discutidas questões de segurança associadas com *cloud brokerage* e apresentada uma arquitetura de *framework* capaz de regular serviços em *cloud bursting*. Em [Tordsson et al. 2012] é apresentado um *broker* que otimiza a alocação de recursos sobre múltiplas nuvens. Similarmente, em [Leitner et al. 2012] é apresentada uma estratégia para computar o escalonamento de requisições sobre nuvens que tenta diminuir a violação do SLA. Em [Shen et al. 2013], é proposta uma estratégia híbrida de escalonamento que minimiza o custo do aluguel por fazer uso instâncias OD e RE. Uma PLI é formulada para realizar o escalonamento. Os três trabalhos não realizam o escalonamento multiusuário e nem tratam questões de segurança no SLA.

Considerando o aspecto de segurança, Marcon et al. [Marcon et al. 2013] introduziram uma estratégia de alocação que aumenta a segurança dos recursos de rede compartilhados entre várias aplicações. Eles agrupam requisições de usuários confiáveis na mesma infraestrutura virtual. Uma PLI computa o escalonamento onde segurança e isolamento são obtidos por alocar cada grupo sobre diferentes infraestruturas virtuais. Diferencia-se do nosso trabalho, pois consideramos que não existe comunicação entre as VMs.

Utilizando heurísticas, Assunção et al. [Assunção et al. 2010] investigaram sete estratégias de escalonamento que consideram o uso de recursos de provedores de IaaS para expandir a capacidade computacional da infraestrutura local para reduzir o tempo de respostas das requisições dos usuários. Similarmente, Le et al. [Le et al. 2013] propuseram uma política adaptativa de gerenciamento de recursos que trata o término de execução das aplicações. Genaud e Gossa [Genaud and Gossa 2011] descreveram doze estratégias baseadas em heurísticas para o escalonamento *on-line* com o objetivo de minimizar o tempo e o custo do escalonamento. Os autores simularam as estratégias sobre um grid.

Em [Casalicchio and Silvestri 2012] as características de disponibilidade, desempenho e segurança são tratadas durante o escalonamento, contudo, sob o ponto de vista do provedor que objetiva aumentar o lucro obtido ao alugar os recursos.

Vieira et al. [Vieira et al. 2015] propuseram uma PaaS que oferece o serviço de escalonamento de requisições de usuário sobre diferentes provedores de IaaS considerando um SLA com duas características: confiabilidade e desempenho. Contudo, trata requisições de apenas um usuário. Os trabalhos existentes na literatura consideram apenas algumas das necessidades de QoS dos usuários separadamente, não o fazem em um único SLA. Este trabalho objetiva preencher esta lacuna ao passo que propõe uma PaaS que considera três características das necessidades do usuário (confiabilidade, desempenho e segurança) em um SLA. Além disso, realiza o escalonamento de requisições pertencentes a vários usuários ao mesmo tempo objetivando o menor custo monetário.

3. Caracterização das Necessidades dos Usuários e a Formulação do Problema

Usuários da web têm utilizado os serviços de computação em nuvem para diminuir os custos monetários com infraestrutura. Os usuários solicitam VMs de provedores de IaaS para executar suas aplicações. Desta forma, definimos a solicitação de uma instância de

VM como *requisição de VM*. Um usuário pode solicitar várias VMs com diferentes níveis de requisitos de QoS. Por outro lado, os provedores utilizam SLAs para especificar as características dos serviços oferecidos em termos de métricas que devem ser aceitas pelas partes, e define as penalidades para as violações do acordo. Existem vários parâmetros importantes em um SLA administrado pelo provedor de IaaS, tais como: tempo, custo, confiabilidade, segurança, localização, desempenho, comunicação, e armazenamento.

Neste trabalho, consideramos três parâmetros no SLA: confiabilidade, desempenho e segurança. Assim, definimos a requisição de uma instância de VM como $r = \{d, d\alpha, qos_c, qos_d, qos_s\}$, onde $r(d)$ representa o total de tempo que a instância deve ficar disponível para o usuário; $r(d\alpha)$ representa um relaxamento sobre d; $r(qos_c)$, $r(qos_d)$ e $r(qos_s)$ representam, respectivamente, as necessidades de confiabilidade; desempenho; e segurança. Denotamos como $\mathcal{R}$ o conjunto de requisições de um usuário. Uma VM é definida como $\psi = \{qos_c, qos_d, c\}$, onde $\psi(qos_c)$ é a confiabilidade da VM; $\psi(qos_d)$ é o desempenho da VM; e c é o custo por alugar a VM por uma unidade de tempo. A PaaS gerencia um conjunto $|\Psi| = m$ de instâncias de VMs alugadas de provedores de IaaS. Descrevemos uma modelagem dos parâmetros confiabilidade e desempenho que deriva da proposta em [Vieira et al. 2015] como características da necessidade do usuário. Uma modelagem do parâmetro de segurança é proposta na Seção 4.

3.1. Modelagem dos Requisitos de Confiabilidade

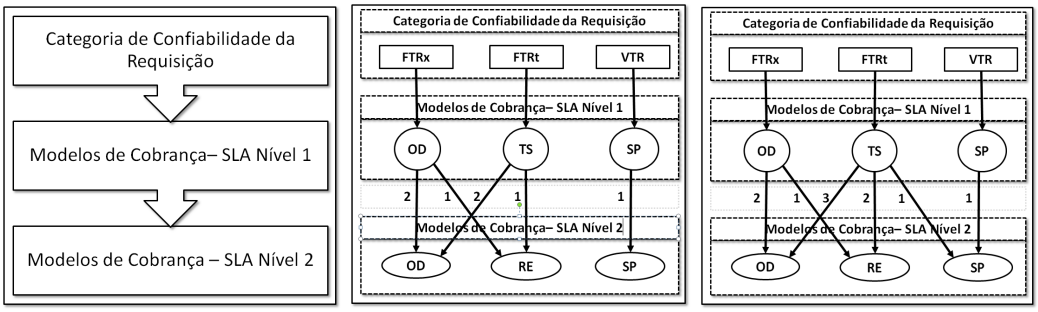
A confiabilidade define se um serviço pode sofrer atraso e/ou ser interrompido. Considerando o tempo de execução e a disponibilidade, classificamos o nível de confiabilidade de requisição do usuário em três categorias. Na primeira categoria, denominada Requisição de Tempo Fixo (FTRx), a requisição deve iniciar imediatamente e a VM não pode ser interrompida (preemptada) durante a execução. Na segunda, denominada Requisição de Tempo Flutuante (FTRt), a requisição pode não iniciar imediatamente. Contudo, uma vez iniciada não pode ser interrompida. E finalmente, na terceira categoria, denominada Requisição de Tempo Variável (VTR), a requisição pode não iniciar imediatamente e pode ser interrompida. Isto é, pode ser fragmentada em requisições menores. Por meio da classificação descrita é possível desenvolver um escalonamento que atenda com mais qualidade as necessidades do usuário referentes ao tempo de execução e à disponibilidade.

3.1.1. Modelo Conceitual e Duas Possibilidades de Mapeamento

Para a plataforma suportar a diversidade existente entre as necessidades do usuário e as características do serviço oferecido pelos provedores, propomos um modelo conceitual de interação entre estas informações (Fig. 1-a). O modelo conceitual possui dois níveis de SLA. O SLA de nível 1 corresponde ao SLA da plataforma e o SLA de nível 2 corresponde ao SLA pertencente aos provedores, respectivamente, denotados por Ω^1 e Ω^2. O nível de categorias de confiabilidade contém as três categorias de confiabilidade descritas anteriormente (FTRx, FTRt e VTR), as quais são mapeadas para algum modelo de cobrança no SLA de nível 1 ($\Omega^1 = \{OD, TS^7$ e $SP\}$) pertencente à plataforma. Em

[7]Modelo *Time Slotted Reservation* (TS) . Neste modelo, o usuário solicita a utilização de uma VM em janelas de tempo [Vieira et al. 2014], e a execução não pode se interrompida. Isto permite utilizar melhores os recursos de instâncias RE, minimizando o curso por unidade de tempo.

seguida, é mapeado para um modelo de cobrança ($\Omega^2 = \{OD, RE \text{ e } SP\}$) de algum provedor público no SLA de nível 2. Desta forma temos a interação entre o usuário e a plataforma e em seguida, a interação entre a plataforma e os provedores de IaaS. Podemos incluir novas características na plataforma sem influenciar diretamente no SLA de nível 2 do modelo conceitual. Além disso, permite que novos modelos de cobrança sejam propostos sem interferir nos modelos atualmente disponíveis nos provedores de IaaS.



(a) (b) (c)

Figura 1. a) Modelo conceitual de mapeamento, b) Mapeamento Conservador, e c) Mapeamento Flexível (Adaptado de [Vieira et al. 2014])

A maneira com a qual o modelo conceitual foi proposta, em níveis, permite que uma série de mapeamentos seja possível. A Fig. 1(b) apresenta uma possibilidade de mapeamento denominada Mapeamento Conservador (MC). Sejam C_{od}, C_{re} e C_{sp}, respectivamente, o custo de uma instância, OD, RE e SP. Temos $C_{od} > C_{re} > C_{sp}$. Desta forma, a prioridade de utilização das instâncias tendem a ser SP, RE e OD, nesta ordem.

Outra possibilidade de mapeamento, denotada como Mapeamento Flexível (MF), Fig. 1(c), utiliza instâncias SP para diminuir o custo do escalonamento. Isso introduz uma potencial violação na QoS da requisição, pois o provedor pode interromper a VM SP. Para evitar isso, a plataforma gerencia duas zonas de VMs alugadas: Zona de Execução (ZE) e Zona de Redundância (ZR). A ZE contém as VMs OD, RE e SP pertencentes ao conjunto Ψ, enquanto que a ZR é composta por m' VM RE no conjunto Ψ'. Para cada requisição r, na qual $r(qos_r) = FTRt$, alocada em uma VM SP na ZE, é criada uma cópia de r denominada r' na ZR que será utilizada se r falhar. O grau de concorrência σ^c é a quantidade de requisições que podem executar concorrentemente em uma VM em Ψ'.

3.2. Modelagem dos Requisitos de Desempenho

Consideramos que o desempenho de uma requisição do usuário é a configuração de hardware necessária para executar a requisição durante um certo tempo de duração. Por outro lado, os provedores de IaaS oferecem diferentes VMs com diferentes características de desempenho. Seja ρ uma métrica de desempenho, $1\rho < 2\rho < ... < n\rho$. Tanto a requisição do usuário quanto a VM oferecida pelo provedor possuem o desempenho qos_d associado. Desta forma, mapeamos a requisição para a instância de VM utilizando a seguinte regra: $r \rightarrow \psi$, se $r(qos_p) = \psi(qos_p)$. Esta não é uma condição suficiente para que a requisição seja escalonada, pois a confiabilidade e a segurança devem ser consideradas também.

3.3. Formulação do Problema

Muitas plataformas têm sido propostas para computar o escalonamento e ajudar os usuários a escolherem os melhores recursos para executar suas aplicações. O nível de

QoS especificado pelo usuário na requisição deve ser considerado durante o processo de escalonamento. A PaaS proposta em [Vieira et al. 2015] inovou ao considerar os requisitos de confiabilidade e desempenho durante o escalonamento. Neste artigo, propomos a extensão da PaaS para incluir o requisito de segurança no SLA. Isso possibilita o escalonamento de requisições de vários usuários ao mesmo tempo. O usuário u_i submete um conjunto $\mathcal{R}_i$ de requisições para a PaaS. O número de usuários é denotado como n_u. Definimos como $\mathcal{R}^+ = \bigcup \mathcal{R}_i$, $1 \leq i \leq n_u$, o conjunto de todas as requisições. A partir disso, formulamos nosso problema como segue:

Compute um escalonamento S de um conjunto $\mathcal{R}^+$ de requisições de VMs sobre um conjunto Ψ de instâncias de VMs de provedores de IaaS objetivando o menor custo de alocação sem violar a QoS das requisições.

A PaaS computa o escalonamento $S = \{t; q; c; E\}$, de $\mathcal{R}^+$, onde: t representa o tempo total de execução de $\mathcal{R}^+$; q representa o número de requisições atendidas; c representa o menor custo para o usuário; e E representa o escalonamento contendo o tempo inicial de cada requisição alocada e sua respectiva VM. Por considerar vários usuários ao mesmo tempo durante o escalonamento, um novo problema é identificado: *Como tratar as características de segurança em um escalonamento de usuários distintos?*

Para contornar este problema propomos o isolamento e compartilhamento de requisições confiáveis e incluímos a segurança como um parâmetro no SLA. Uma estratégia ingênua de escalonamento pode elevar o custo monetário. Portanto, é importante produzir bons escalonamentos utilizando algoritmos mais elaborados. Alocação de recursos baseada em SLA com diferentes parâmetros em computação em nuvem é um problema NP-Difícil [Bittencourt et al. 2012].

4. Um Escalonamento Multiusuário Sobre Múltiplos Provedores de IaaS

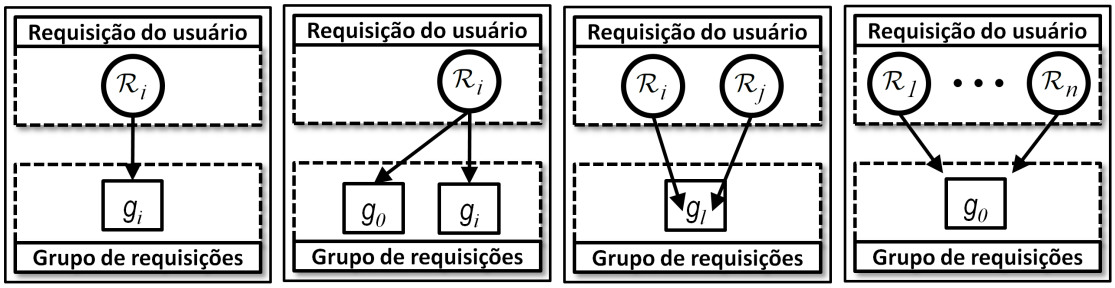
Propomos uma modelagem do requisito de segurança para estender o SLA proposto em [Vieira et al. 2015] para um SLA tri-dimensional que considera a confiabilidade, o desempenho e a segurança como dimensões. Apresentamos um algoritmo para computar o escalonamento utilizando o SLA proposto.

4.1. Modelagem dos Requisitos de Segurança

Propomos a geração de grupos de isolamento e de compartilhamento de requisições entre usuários confiáveis. Apresentamos quatro possibilidades de geração de grupos (Fig. 2), descritas a seguir:

(a) Isolamento completo: Requisições entre usuários distintos não são executadas na mesma VM. Um grupo g_i é criado para as requisições $\mathcal{R}_i$ de cada usuário, $1 \leq i \leq n_u$;
(b) Isolamento parcial: Parte das requisições de usuários distintos são atribuídas a um mesmo grupo (g_0) e podem ser executadas na mesma VM. Contudo, uma outra parte das requisições continuam isoladas completamente;
(c) Compartilhamento entre grupos confiáveis: Um grupo para mais de um usuário, possibilitando o compartilhamento de grupos de requisições de usuários confiáveis.
(d) Compartilhamento completo: Uma requisição pode ser executada com qualquer outra. Apenas um grupo é criado utilizando este modelo de geração de grupos.

Outras possibilidades podem ser incluídas como mapeamento para que o escalonamento atenda às necessidades dos usuários da plataforma.



(a) (b) (c) (d)

Figura 2. Possibilidades de modelos de geração de grupos

4.2. Um Algoritmo para Computar o Escalonamento

Propomos um algoritmo em quatro fases. O Algoritmo 1 apresenta a estratégia proposta.

Algoritmo 1 Escalonamento Multi-Usuário

Entrada: $\mathcal{R}^+$, Ψ, σ^c, m^g, m^c
Saída: S

1: $\mathcal{R}^+_{\mathcal{G}} \leftarrow$ *Geração dos grupos de isolamento*($\mathcal{R}^+$, m^g) {Primeira Fase}
2: $\mathcal{R}_{SLA^2} \leftarrow$ *Mapeamento das categorias de cofiabilidade*($\mathcal{R}^+_{\mathcal{G}}$, m^c) {Segunda Fase}
3: $S_{ze} \leftarrow$ *Computação da zona de execução*($\mathcal{R}_{SLA^2}$, Ψ) {Terceira Fase}
4: $S_{zr} \leftarrow$ *Computação da zona de redundância*(S_{ze}, σ^c) {Quarta Fase}
5: Compute $S \leftarrow S_{ze} + S_{zr}$

O modelo para geração de grupos de requisições m^g (Fig. 2) é utilizado para geração dos grupos (primeira fase). O modelo de mapeamento das categorias de confiabilidade m^c é utilizado para o mapeamento das categorias (segunda fase). Utilizamos uma PLI para computar a zona de execução (terceira fase) e uma heurística baseada no grau de utilização das VMs para computar a zona de redundância (quarta fase).

4.3. Primeira Fase: Geração dos Grupos de Isolamento

Realizamos o agrupamento de requisições de usuários confiáveis em grupos do conjunto $\mathcal{G} = \{g_0, g_1, g_2, ..., g_{n_g}\}$. Denotamos n_g como o número de grupos criados. A primeira fase do Algoritmo 1 é computada utilizando um dos modelos apresentados na Fig. 2.

4.4. Segunda Fase: Mapeamento das Categorias de Confiabilidade

Durante a segunda fase, realizamos o mapeamento das requisições dos usuários seguindo o modelo especificado em m^c. Atualmente, a plataforma disponibiliza dois tipos de mapeamento como descrito anteriormente: mapeamento conservador e mapeamento flexível.

4.5. Terceira Fase: Computação da Zona de Execução

Para computar o escalonamento da ZE, incluindo os grupos de isolamento e de compartilhamento de requisições, introduzimos duas importantes modificações na PLI proposta em [Vieira et al. 2015]: a primeira está relacionada à inclusão da terceira dimensão no SLA utilizando a geração de grupos computados na primeira fase do Algoritmo 1, a segunda modificação está relacionada a permitir mais de uma requisição ser executada ao

mesmo tempo na mesma VM. A PLI computa o escalonamento na terceira fase utilizando as variáveis binárias u, v, w, x, y, e z e as constantes C, $\mathcal{B}$ e $\mathcal{K}$ como a seguir:

- $u_{t,\psi}$: Assume o valor 1 se ψ executa alguma requisição no instante t. Senão, 0;
- $v_{g,\psi}$: Assume o valor 1 se ψ executa alguma requisição do grupo g. Senão, 0;
- w_r: Assume o valor 1 se a requisição r é executada. Senão, 0;
- $x_{r,\psi}$: Assume o valor 1 se r é executada em ψ. Senão, 0;
- $y_{r,t,\psi}$: Assume o valor 1 se r é executada em ψ no instante t. Senão, 0;
- $z_{r,t,\psi}$: Assume o valor 1 se a requisição r com $r(qos_c) = \{FTRt\}$ inicia a execução no instante t sobre a VM ψ. Senão, 0;
- $C_{t,\psi}$: Constante que assume o custo por unidade de tempo por utilizarmos a VM.
- $\mathcal{B}$: Constante suficientemente grande que assume um peso para cada requisição.
- $\mathcal{K}$: Constante suficientemente grande utilizada para garantir que requisições FTRt iniciem apenas uma vez.

Formulamos a função objetivo $F = \sum_{t \in T} \sum_{\psi \in \Psi} (u_{t,\psi} \times C_{t,\psi}) - \sum_{r \in \mathcal{R}} (w_r \times \mathcal{B})$ que computa o escalonamento de $\mathcal{R}^+$ sobre Ψ objetivando o menor custo de alocação sem violar o QoS da requisição. Devemos minimizar F sujeito às seguintes restrições:

$$\sum_{t=1}^{r(d\alpha)} y_{r,t,\psi} = r(d) \times x_{r,\psi}; \forall r \in \mathcal{R}, \forall \psi \in \Psi \quad (1)$$

$$\sum_{g \in G} v_{g,\psi} \leq 1; \forall \psi \in \Psi \quad (2)$$

$$x_{r,\psi} \leq v_{r(g),\psi}; \forall \psi \in \Psi, \forall r \in \mathcal{R} \quad (3)$$

$$\sum_{\psi \in \Psi} x_{r,\psi} = w_r; \forall r \in \mathcal{R} \quad (4)$$

$$\sum_{r \in \mathcal{R}} (y_{r,t,\psi} \times r(qos_d)) = u_{t,\psi} \times \psi(k); \forall t \in \left[1, r(d\alpha)\right], \forall \psi \in \Psi \quad (5)$$

$$\sum_{r \in R} y_{r,t,\psi} = 0; \forall t \in T; \psi \in \Psi; MF(r,\psi) = 0 \quad (6)$$

$$v_{g,\psi}, w_r, x_{r,\psi}, y_{r,t,\psi}, z_{r,t,\psi} \in \{0,1\}; \forall g \in G, \forall r \in \mathcal{R}, \forall t \in \mathcal{T}, \forall \psi \in \Psi \quad (7)$$

$$\sum_{t=1}^{\mathcal{L}} z_{r,t,\psi} = x_{r,\psi}; \forall r \in \mathcal{R}, \forall \psi \in (\Psi^{od} \bigcup \Psi^{re}) \quad (8)$$

$$r(d) - \mathcal{K} \times (1 - z_{r,t,\psi}) \leq \sum_{s=t}^{r(d)+t-1} y_{r,s,\psi} \leq r(d) + \mathcal{K} \times (1 - z_{r,t,\psi}) \quad (9)$$

$$\forall r \in \mathcal{R}, \forall t \in \left[1, r(d\alpha) - r(d) + 1\right], \forall \psi \in \Psi, r(qos_c) = \{FTRx, FTRt\}$$

A restrição (1) especifica que se uma requisição for executada, ela deve ser executada em $r(d)$ unidades de tempo e em uma única VM. As restrições (2) e (3) trabalham juntas e especificam que as requisições escalonadas em uma VM devem pertencer

ao mesmo grupo de isolamento. A primeira especifica que uma VM ψ deve atender requisições de no máximo um grupo. A segunda especifica que se a requisição r executar na VM ψ, a VM ψ deve executar somente requisições do grupo da requisição de r, denotado como $r(g)$. A restrição (4) especifica que se uma requisição for escalonada, ela contribuirá para o custo total.

A restrição (5) especifica que a soma do desempenho das requisições executadas em uma unidade de tempo deve ser igual ao desempenho da instância de VM na qual as requisições foram alocadas. A garantia de execução conforme o modelo de mapeamento entre SLA de nível 1 e SLA de nível 2 apresentado na Fig. 1 é feita pela restrição 6) em conjunto com a função de mapeamento FM(). A restrição (7) assegura que as variáveis u, v, w, x, y e z sejam binárias. As restrições (8) e (9) asseguram que requisições do tipo FTRt sejam executadas de maneira atômica.

A PLI computa o melhor escalonamento com o maior número de requisições atendidas. Uma requisição não pode ser atendida parcialmente. O valor da constante $\mathcal{B}$ é atribuído para cada requisição w_r atendida. Seja C_p o custo computado pela PLI. Então, $S.q = \lfloor \frac{-C_p}{\mathcal{B}} \rfloor + 1$ e $S.c = (S.q \times \mathcal{B}) - C_p$. Seja d_m e ψ_m, respectivamente, o maior tempo de execução entre todas as requisições $r \in \mathcal{R}^+$ e o maior custo por unidade de tempo entre todas as VMs $\psi \in \Psi$. Deste modo, $S.q$ e $S.t$ podem ser computados como descrito anteriormente somente se $\mathcal{B} > (d_m \times \psi_m)$. Esta é a maneira como $\mathcal{B}$ deve ser definida. Caso contrário, o resultado da PLI pode ser comprometido.

4.6. Quarta Fase: Computação da Zona de Redundância

Quando a terceira fase termina, o escalonamento computado pode ter gerado resultados onde requisições FTRt são escalonadas em VMs SP. Como descrito anteriormente, requisições FTRt não podem ser preemptadas e podem sofrer um atraso inicial estabelecido por $r(d\alpha)$. Contudo, VMs SP podem ser preemptadas e deste modo precisamos alocar estas requisições para VMs na zona de redundância, então elas podem ser ativadas no caso das VMs SP serem preemptadas na ZE. Definimos $\Psi^\circ = \{\psi_1, \psi_2, ..., \psi_{m^\circ}\}$ como o conjunto de VMs nas quais ao menos uma requisição FTRt tenha sido alocada. Quando a VM pertence ao grupo de isolamento g, utilizamos Ψ°_g. Então, na quarta fase do Algoritmo 1 devemos computar a ZR para as requisições em Ψ°.

Em [Vieira et al. 2014], propusemos o escalonamento Baseado no Grau de Utilização (BGU) para computar a ZR considerando o grau de utilização da VM no escalonamento da ZE. Neste trabalho apresentamos o Algoritmo 2, denominado *Grau de Utilização 3D* (GU-3D), para computar a ZR. O algoritmo é baseado no Algoritmo BGU, contudo, considera 3 dimensões no SLA: confiabilidade, desempenho e segurança.

Definimos o *grau de utilização* σ^u_ψ como a quantidade de unidades de tempo que uma VM ψ executa uma requisição de acordo com o escalonamento S_{ze} da zona de execução. Por exemplo, suponha que somente as requisições r_1 e r_2 do tipo FTRt com $r_1(d) = 25$ e $r_2(d) = 40$ foram escalonadas, r_1 antes de r_2, para executar na VM ψ_i em S_{ez}. Neste caso, $\sigma^u_{\psi_i} = 65$. A ideia básica do Algoritmo 2 é computar $\sigma^u_\psi, \forall\ \psi \in \Psi^\circ_g$ e então utilizar a ordem dada por σ^u para escalonar ψ para ψ', cada uma contendo σ^c VMs. Isto deverá ser feito de acordo com o modelo de geração de grupos especificado.

Na linha 1 computamos Ψ°_g, $g \in \mathcal{G}$, os grupos de isolamento de acordo com o modelo de grupos definido em m^g. O grau de utilização σ^u para cada VM ψ pertencente

Algoritmo 2 Grau de Utilização 3D (GU-3D)

Entrada: S_{ze}, σ^c, m^g
Saída: Schedule S_{rz}

1: **para cada** $g \in \mathcal{G}$ **faça**
2: $\Psi_g^\circ \leftarrow$ IGG(Ψ°, m^g);{Compute os grupos de isolamento}
3: **final para**
4: **para cada** $g \in \mathcal{G}$ **faça**
5: **para cada** $\psi \in \Psi_g^\circ$ **faça**
6: Compute σ_ψ^u;{Compute o grau de utilização de S_{ze}}
7: **final para**
8: **final para**
9: **para cada** $g \in \mathcal{G}$ **faça**
10: $\Psi_g^\circ \leftarrow$ Orderne Ψ_g° por σ^u
11: $m_g \leftarrow |\Psi_g^\circ|$
12: **para** $i = 1$ até m_g **faça**
13: $\psi'_{\lceil \frac{i}{\sigma^c} \rceil} \leftarrow \psi'_{\lceil \frac{i}{\sigma^c} \rceil} \bigcup \psi_g^i$ {Compute o escalonamento}
14: **final para**
15: **final para**
16: $S_{zr}.c$;{Compute o custo do escalonamento}
17: retorne S_{zr}

a cada grupo de isolamento é computado na linha 4. Então, ordenamos o conjunto Ψ_g° por σ^u para cada grupo de isolamento na linha 10 e computamos o escalonamento na linha 12 utilizando a ordenação. O custo monetário do escalonamento é dado pela soma das unidades de tempo na qual alguma VM ψ' tem uma requisição alocada para ela, multiplicada pelo custo de cada unidade de tempo. Então, na linha 16, computamos o custo do escalonamento $S_{rz}.c$. Denotamos $\psi'.t$ como uma unidade de tempo de uma VM que assume o valor 1 se ψ_i está executando uma requisição no tempo t. Assume o valor zero caso contrário. Deste modo, o valor de $S_{rz}.c$ é computado como: $S_{rz}.c = \sum \psi'.t \times C_{re}; \quad \forall\ \psi' \in \Psi', \quad 0 < t \leq T$.

A complexidade assintótica do Algoritmo 2 é dominada pela computação do grau de utilização σ^u e pela ordenação do conjunto Ψ°, resultando em $O(m^\circ \times T + m^\circ \log m^\circ)$, onde T é o tempo total considerado para a disponibilidade das VMs. Embora tenha uma complexidade polinomial, utilizando uma heurística, ele não garante a qualidade da solução. Isso acontece pelo fato de que ele considera apenas a quantidade de unidades de tempo utilizadas em uma VM, e não o instante de utilização.

Se uma preempção ocorrer, a concorrência na zona de redundância é imediatamente interrompida, deixando a cópia da requisição que foi preemptada na zona de execução concluir sua execução na zona de redundância sozinha permitindo que ela termine dentro do prazo do relaxamento $r(d\alpha)$ sem violar a QoS. Observe que a preempção de duas ou mais requisições na zona de execução que compartilham a mesma VM na zona de redundância causará falha à segunda (ou as posteriores) requisição preemptada, pelo fato de já ter sido excluída da zona de redundância. Denotamos C_1 e C_2, respectivamente, como os custos dos escalonamentos computados na terceira e na quarta fase. Além disso, denotamos C_3 como o custo para tratar a preempção das VMs durante a execução da

requisição. O custo C_3 é computado como $C_3 = \tau_{eer} \times C_{re} - \tau_{dp} \times C_{sp}$, onde, C_{re} e C_{sp} denotam o custo de utilização por unidade de tempo de uma instância de VM, respectivamente, do tipo RE na zona de redundância e do tipo SP na zona de execução. É necessário computar um tempo extra τ_{eer} e um desconto τ_{dp} pela VM interrompida na zona de execução, pois o custo já havia sido computado durante a terceira fase.

Denotamos o custo total do escalonamento C_t como $C_t = C_1 + C_2 + C_3$. Se não ocorrem preempções, temos $C_3 = 0$. Para que o escalonamento tenha um resultado satisfatório para o usuário, é importante que ele adéque inteligentemente às necessidades dos usuários. O SLA com três dimensões proposto contribui para este objetivo.

5. Resultados

Implementamos em Java o Algoritmo 1 para computar o escalonamento. As métricas observadas são o custo do escalonamento, a quantidade de requisições atendidas e a quantidade de VMs de cada tipo (OD, RE e SP) utilizadas. Utilizamos o IBM ILOG CPLEX com o limite de 90 minutos para cada execução da PLI. Realizamos 10 experimentos, E_i, $1 \leq i \leq 10$, cada um com 30 conjuntos de requisições, $E_i = \{R_1, R_2, ..., R_{30}\}$. Em cada experimento, o número de requisições em cada conjunto é E_i: $|R_j| = i \times 10$, $1 \leq j \leq 30$. Além disso, configuramos $r(d) \leftarrow$ *random*(1,20) e $r(d\alpha) \leftarrow$ *random*(1, 80)+$r(d)$ de cada requisição em cada conjunto $R_j \in E_i$. Como apresentado na Tabela 1 temos recursos disponíveis de dois provedores. O custo de cada VM é proporcional, multiplicado por 100, ao custo utilizado pela Amazon EC2 atualmente. O conjunto $\mathcal{R}$ é composto de requisições na proporção de de 20% (FTRx), 60%(FTRt) e 20%(VTR). Avaliamos diferentes graus de concorrência $\sigma^c = \{2, 4, 8\}$ na zona de redundância. Além disso, todas as requisições necessitam de apenas uma unidade de processamento (1ρ).

Tabela 1. Conjunto de VMs disponíveis para a plataforma.

Provedor	Desempenho	OD	$	RE	$	SP	$	TOTAL
1	1ρ	1	70.00	-	-	-	-	2
	2ρ	1	140.00	-	-	-	-	
2	1ρ	-	-	1	40.00	6	8,00	14
	2ρ	-	-	1	100.00	6	16,00	
							Total	**16**

Utilizamos a PLI e o Algoritmo 2 para computar, respectivamente, a zona de execução e a zona de redundância durante a execução dos 10 experimentos. Os resultados evidenciam o potencial da utilização das VMs do tipo SP no escalonamento. As Figs. 3(a), 3(b), 3(c) e 3(d) apresentam o custo do escalonamento obtido utilizando os mapeamentos conservador e flexível considerando, respectivamente, $n_u = 1$, $n_u = 2$, $n_u = 3$, e $n_u = 4$. A curva denotada por MC apresenta os resultados obtidos pela execução dos experimentos utilizando o MC. A curva denotada por ZE apresenta os resultados do escalonamento da zona de execução utilizando o MF.

Analisando os dados, observamos que para $n_u = 1$ (Fig. 3-a), a estratégia de utilizar o mapeamento flexível com $\sigma^c = 2$ apresenta um custo mais baixo comparado ao custo obtido pelo mapeamento conservador, exceto no Experimento E_{10}. Pois, neste experimento, temos muitas requisições e o custo de computar a zona de redundância é alto. Um comportamento semelhante pode ser observado para $n_u = 2$, (Fig. 3-b). Contudo, não

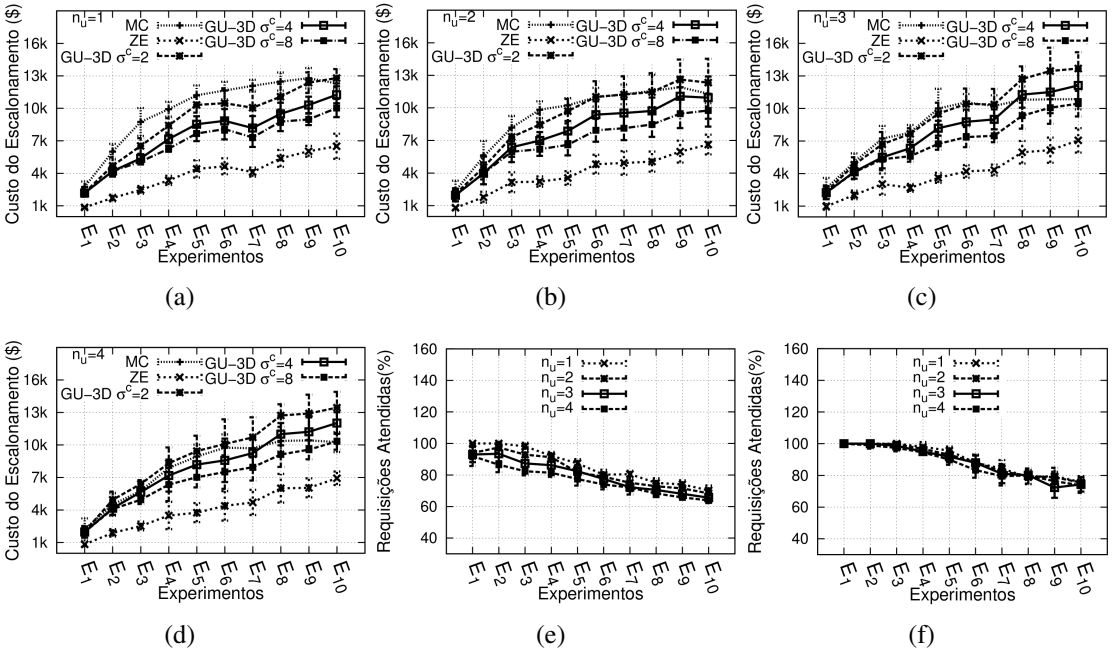


Figura 3. Custo do escalonamento e quantidade de requisições atendidas com $n_u = \{1, 2, 3, 4\}$.

é vantagem utilizar o mapeamento flexível com $\sigma^c = 2$ nos experimentos E_6, E_7, E_8, E_9 e E_{10}. No cenário com quatro usuários (Fig. 3-d), todos os experimentos com $\sigma^c = 8$ obtiveram um custo menor comparado ao obtido pelo escalonamento utilizando o mapeamento conservador. Por outro lado, utilizando $\sigma^c = 2$ todos os experimentos tiveram o escalonamento com custo superior comparado ao obtido pelo escalonamento conservador.

As Figs. 3(e) e 3(f) apresentam a quantidade de requisições atendidas utilizando, respectivamente, os modelos de mapeamento conservador e flexível. Podemos observar que, embora o escalonamento para apenas um usuário (n_1) tenha um custo maior, são atendidas mais requisições, quando comparado aos resultados do escalonamento para n_2, n_3 e n_4. Isso justifica a custo mais alto. Para $n_u = 4$, temos o menor custo. Contudo, a quantidade de requisições atendidas é menor.

A Fig. 4 apresenta a quantidade de VMs de cada tipo considerando os mapeamentos conservador em a), b) e c) e flexível em d), e) e f). Nos Experimentos E_1, E_2, E_3 e E_4, verificamos uma diferença quando comparamos a quantidade de VMs OD utilizada no mapeamento conservador (Fig. 4-a) e no mapeamento flexível(Fig. 4-d) . O mapeamento flexível utiliza menos requisições OD. Para o Experimento E_1, foram utilizadas aproximadamente 10% das VMs no mapeamento flexível e 41% no mapeamento conservador para $n_u = 1$. Esta diferença tem um impacto no custo, pois as VMs OD têm o maior custo. A quantidade de VMs do tipo RE é semelhante nos dois mapeamentos, Fig. 4(b) e Fig. 4(e), atingindo 100% de utilização no Experimento E_2 em diante. Por outro lado, a quantidade de VMs do tipo SP utilizada no mapeamento conservador, Fig. 4(c), é inferior à utilizada no mapeamento flexível, Figura 4(f). Desta forma, observamos que o mapeamento conservador utiliza mais VMs do tipo OD do que o mapeamento flexível; e menos VMs do tipo SP do que o mapeamento flexível. Este comportamento contribui para que o escalonamento computado utilizando o mapeamento flexível tenha um custo menor.

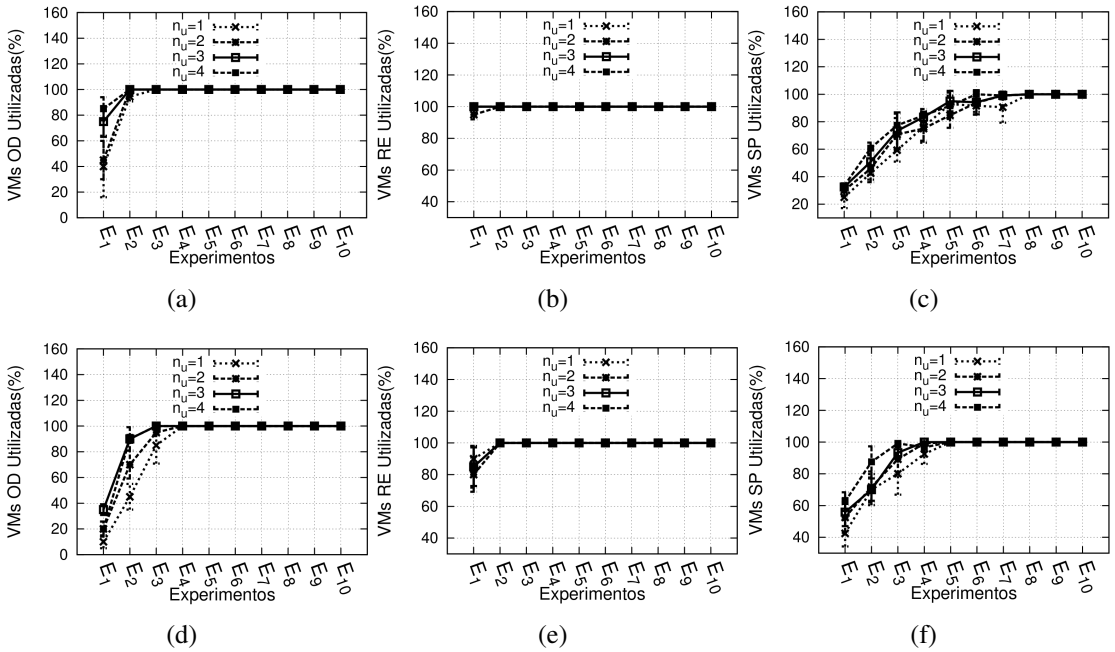


Figura 4. Quantidade de VMs utilizadas nos Mapeamento Conservador e Flexível

Observamos nos resultados dos experimentos realizados que quando consideramos requisições de usuários distintos e incluímos os grupos de isolamento, o custo do escalonamento aumenta. Isso ocorre pelo fato de termos que utilizar mais VMs para atender requisições em diferentes grupos que não podem compartilhar recursos. Esta é uma consequência por aumentar a segurança entre os usuários da PaaS.

6. Conclusão e Trabalhos Futuros

A grande diversidade de serviços e de provedores de IaaS dificulta a escolha dos melhores recursos para os consumidores implantarem suas aplicações. Considerar as características das necessidades do usuário é fundamental para escolher os serviços de maneira eficiente e reduzir os custos. Diante disso, apresentamos uma estratégia de escalonamento que utiliza um SLA com os parâmetros: confiabilidade, desempenho e segurança. Utilizando este SLA é possível considerar requisições de vários usuários no escalonamento além de se adequar melhor às necessidades do usuário enquanto diminui o custo do escalonamento e evita a violação de QoS. Os resultados experimentais mostram que ao introduzir a segurança no SLA, o custo do escalonamento pode aumentar. Contudo, possibilita que diferentes usuários tenham diferentes políticas de segurança. Como trabalho futuro, pretendemos introduzir o escalonamento *on-line* de requisições de diferentes usuários utilizando este SLA possibilitando que as requisições sejam modificadas ao longo do tempo.

Agradecimentos

Este trabalho foi parcialmente financiado pela CAPES e pelo CNPQ.

Referências

Assunção, M. D., Costanzo, A., and Buyya, R. (2010). A cost-benefit analysis of using cloud computing to extend the capacity of clusters. *Cluster Computing*, 13(3):335–347.

Bittencourt, L. F., Madeira, E. R. M., and da Fonseca, N. L. S. (2012). Scheduling in hybrid clouds. *Communications Magazine, IEEE*, 50(9):42–47.

Casalicchio, E. and Silvestri, L. (2012). An inter-cloud outsourcing model to scale performance, availability and security. In *Proceedings of 5th IEEE/ACM International Conference on Utility and Cloud Computing*, UCC 2012, Chicago, USA.

Genaud, S. and Gossa, J. (2011). Cost-wait trade-offs in client-side resource provisioning with elastic clouds. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 1–8.

Le, G., Xu, K., and Song, J. (2013). Dynamic resource provisioning and scheduling with deadline constraint in elastic cloud. In *Service Sciences (ICSS), 2013 International Conference on*, pages 113–117.

Leitner, P., Hummer, W., Satzger, B., Inzinger, C., and Dustdar, S. (2012). Cost-efficient and application sla-aware client side request scheduling in an infrastructure-as-a-service cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 213–220.

Marcon, D., Oliveira, R., Neves, M., Buriol, L., Gaspary, L., and Barcellos, M. (2013). Trust-based grouping for cloud datacenters: Improving security in shared infrastructures. In *IFIP Networking Conference, 2013*, pages 1–9.

Nair, S., Porwal, S., Dimitrakos, T., Ferrer, A., Tordsson, J., Sharif, T., Sheridan, C., Rajarajan, M., and Khan, A. (2010). Towards secure cloud bursting, brokerage and aggregation. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 189–196.

Shen, S., Deng, K., Iosup, A., and Epema, D. (2013). Scheduling jobs in the cloud using on-demand and reserved instances. In *Proceedings of the 19th International Conference on Parallel Processing*, Euro-Par'13, pages 242–254, Berlin, Heidelberg. Springer-Verlag.

Silva, C. A. D. and de Geus, P. L. (2015). Return on security investment for cloud computing: a customer perspective. In *The 7th International Conference on Management of computational and collective IntElligence in Digital EcoSystems (MEDES'15)*, Caraguatatuba-SP, Brazil.

Tordsson, J., Montero, R. S., Moreno-Vozmediano, R., and Llorente, I. M. (2012). Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems*, 28(2):358 – 367.

Vieira, C. C. A., Bittencourt, L. F., and Madeira, E. R. M. (2014). Reducing costs in cloud application execution using redundancy-based scheduling. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 117–126.

Vieira, C. C. A., Bittencourt, L. F., and Madeira, E. R. M. (2015). A two-dimensional sla for services scheduling in multiple iaas cloud providers. *Int. J. Distrib. Syst. Technol.*, 6(4):45–64.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1:7–18.

LB-RLT Approach for Load Balancing Heterogeneous Storage Nodes

Antonio M. R. Almeida, Denis M. Cavalcante, Flávio R. C. Sousa e Javam C. Machado [1]

[1] Mestrado e Doutorado em Ciencia da Computacao (MDCC)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

{manoel.ribeiro,denis.cavalcante}@lsbd.ufc.br {sousa,javam}@ufc.br

Abstract. *Cloud computing is a paradigm of service-oriented computing and has changed the way computing infrastructure is abstracted and used. The cloud is composed by heterogeneous resources and has a variable workload. Thus, load balancing techniques are crucial to distribute workload to processing nodes for enhancing the overall system performance. Conventional algorithms for load balancing have limitations in this environment, or they do not consider specific aspects of the resources simultaneously, e.g., response time and throughput. To address these limitations, this paper presents an approach to load balancing in the cloud. This approach considers cloud infrastructure storage throughput and a heterogeneous storage nodes environment. In order to evaluate this approach, we have conducted experiments that measure the response time, throughput and success rate and compared our results against conventional algorithms. Experimental results confirm that our approach ensures quality of service agreement, while using resources more efficiently.*

1. Introduction

Cloud computing is a paradigm of service-oriented computing and has changed the way infrastructure is abstracted and used. Infrastructure as a service (IaaS) is the most basic cloud model for provisioning compute, storage, and networking resources. In cloud storage, there are major types of storage resources, layered on the following order bottom-up: (a) Block storage: low level I/O access to records through a fixed size (i.e. volumes); (b) File storage: files are composed by its data and are organized through a hierarchical directory. A file is associated to its directory by a well-defined system meta-data that also provides some basic information such as access time, file size etc; (c) Object storage: objects are composed by data and user-accessible attributes (i.e user meta-data) [Mesnier et al. 2003]. User meta-data can be included into objects using any custom key/value, thus making data and user meta-data physically together.

Object storage systems are well suited to scaling-out by replicating the object itself several times for redundancy as well as for responding to a higher demand. During a higher demand, a common necessity of cloud providers is to be able to meet the most strict Quality of Service (QoS) levels defined in a Service Level Agreement (SLA) better using the available resources before allocating more replicas/servers. However, it is still a challenging issue for dynamic application providers how to efficiently tackle scenarios of gradual load variations and load peaks from the workloads seen by dynamic applications.

For this challenge, the proper load balance of replicas are a key point to avoid violations of SLA requirements and to reduce infrastructure costs. [Xie et al. 2015]

Many works have been developed in order to optimize the load balancing of replica placement in systems based on centralized naming such as Lightweight Directory Access Protocol (LDAP) or distributed ones such as distributed hash tables (DHT) (i.e. hash functions) that avoids searching for available nodes in a central directory [Brinkmann et al. 2000] [Allcock et al. 2002] [Godfrey et al. 2004]. HUSH and CHUSH algorithms have proposed hashing functions for reducing look-up response time as well as load balancing the replicas according to server weighting [Honicky and Miller 2004] [Weil et al. 2006].

Another important front of research has focused on load balancing of replica selection, i.e., which replica is the most suitable for being used taking into account some aspects such as performance, cost, security, number of accesses, and replica crashes [Rajalakshmi et al. 2014]. By this means, replica selection is part of load balancing techniques that aim to distribute workload to processing nodes for enhancing the overall performance of system.

Many common aspects of cloud systems leveraged by load balancing approaches have been classified according to the type of algorithm, knowledge base, issues to be addressed, usage and drawbacks [Katyal and Mishra 2013]. Even though some authors have not yet approached fine-grained load balancing by distributing the read workload over data replicas [Weil et al. 2006]. Other authors optimized replica selection by considering disk head scheduling, cache utilization and inter-brick load balancing [Lumb et al. 2003]. It is clear that these load balancing approaches could be considered by replica selection optimization in order to improve some specific aspect of a cloud system.

An ideal load balancing algorithm for homogeneous resources should avoid overloading or under loading of any specific node. However, the cloud is composed by heterogeneous resources and variable workloads, thus making harder the implementation of load balancing strategy because it involves additional processing to efficiently detect how much one replica should be selected more than others while still avoiding starvation. Conventional algorithms for load balancing have limitations in this environment, or they do not consider specific aspects of the resources simultaneously, e.g., response time and throughput. To address these limitations, this paper presents an approach for load balancing replica selection in a cloud object storage. The major contributions of this paper are as follows:

- It proposes an approach to load balance replica selection of read requests for heterogeneous nodes by combining and comparing their last read data throughput and their total of read data.
- It presents a full prototype implementation in a real cloud object storage system, an experimental evaluation and the results showing that our approach is effective in balancing replica selection of read requests for heterogeneous storage nodes environment.

Organization: This paper is organized as follows: Section II explains our approach. The environment is described in Section III. The evaluation of our approach is presented in Section IV. Section V surveys related work and, finally, Section VI presents the conclusions.

2. LB-RLT Approach

Our approach, Load Balance by Round-robin, Last Read Data Chunk Throughput and Total of Read Data Chunks (LB-RLT), distributes dynamically the read requests of objects through node replicas. LB-RLT takes advantage of an architecture that consists of a proxy node accounted for spreading the workload through multiple node replicas. Every node replica is a storage node that is able to read/write one object replica requested or submitted by a proxy node in order to support a client demand for an object, thus making the object access transparently for the client as shown in the Figure 1.

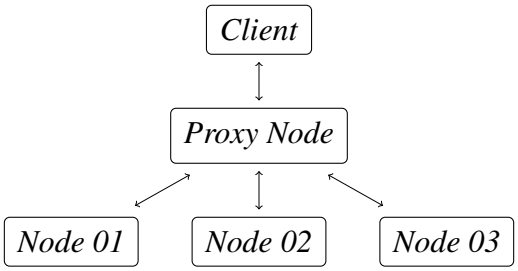


Figure 1. Architecture

To improve understanding, Table 1 shows the definitions. In the Figure 1, we can think about how an object is requested or submitted by the client until it reaches the Node 01, Node 02 or Node 03. It is a common technique for transferring objects to split and transfer the object in small units called dataChunk. These dataChunks have a fixed size, thus the client sends/reads multiples dataChunks to/from the proxy node that forwards/retrieves them until an object is totally transferred. For our replicated system, one replica is sufficient to process a read request.

Table 1. Definitions

Term	Description
DataChunk	A fragment of an object
DataChunkSize	Standard size of a dataChunk
DataChunkTransferTime	Total transfer time in seconds of a dataChunk between a proxy node and a storage node
Throughput	Transfer rate per object or DataChunk (bytes / seconds)
LastReadDataChunkThroughput	The throughput measured by the last reading of a dataChunk from a storage node
TotalReadDataChunks	The number of dataChunks sucessfully read from a storage node until present moment

It is important to note that the transfer time for each dataChunk to be transferred is very sensitive to the network as well as the node performance, i.e., dataChunkTransferTime gets lower when less requests are demanded to a storage node or gets higher when more requests are demanded to a storage node. By this means, we define the Equation 1:

$$Throughput = \frac{dataChunkSize}{dataChunkTransferTime} \quad (1)$$

Considering the previous architecture showed at Figure 1 and the Equation 1 used to measure throughput during the reading of objects, LB-RLT adds to the proxy node the capability of collecting in runtime the following metrics for every storage node:

- *LastReadDataChunkThroughput:* The measurement of a throughput evaluated in the last reading of dataChunk from a storage node by a proxy node as described by the Equation 1. In this manner, the lastReadDataChunkThroughput metric is updated frequently instead of being collected only when an object is totally read. Once the proxy node holds a lastReadDataChunkThroughput for each storage node regardless of which object is being read, every new read object request will consider the last throughput performed for each storage node.
- *TotalReadDataChunks:* The measurement of how many dataChunks were read from each storage until present moment, i.e., totalReadDataChunks metric is incremented only for dataChunks received with success by a proxy node.

By having both metrics being evaluated in runtime and using round-robin sorting to spread equally when the previous metrics are the same, LB-RLT adds to the proxy node, the capability of load balancing the new coming read request by evaluating and comparing the storage nodes using the Equation 2.

$$Performance = lastReadDataChunkThroughput \times totalReadDataChunks \quad (2)$$

Our performance model is able to evaluate the node performance when throughput metric is much higher/lower than the totalReadDataChunks, thus being the important value for distinguishing performance among replica nodes as well as when throughput is relatively similar among the replica nodes. In this second case, the totalReadDataChunks is able to evaluate which storage node has performed more reading requests with success in order to distinguish the best node. Our approach considers different types of workloads according to the following:

- Low/medium workloads: any replica node would respond the requests with success, but the best replica nodes would be more used because of their highest throughput;
- High workloads: best replica nodes would still respond with success, but their throughput would get a little bit down. In this scenario, our model would let the worse replica nodes be chosen a little bit more due the fact that it is still better to use the bad nodes than otherwise;
- Extreme High workloads / Above cloud capacity: any storage node replica would be used to attend the extreme workload stress.

LB-RLT aims to load balance heterogeneous storage node by targeting high throughput, low response time and longer high success rate with no profiling information required. Table 2 summarizes the features of the approach.

2.1. LB-RLT Algorithm

LB-RLT is described by two algorithms: (1) the LB-RLT Performance Score Evaluation Algorithm 1 that updates the performance score of each replica node according to our approach and (2) the LB-RLT Read Object Request Algorithm 2 that sorts the replica nodes using our approach and choose the best one to read the data object.

In Algorithm 1, on line 1, the input parameter replicasNodeList is all the storage nodes that contain a replica of the requested object. On line 2, the output parameter is

Table 2. LB-RLT Approach

Type of Algorithm	Dynamic and Centralized
Prerequisite	Replicated Data
Knowledge Base	Locally and in runtime measured
Issue to be addressed	LastReadDataChunkThroughput, TotalReadDataChunks
Usage	Heterogeneous Nodes / No profiling needed
Measure expiration	yes

the replicaNodeList with its performance score updated for each replica node. On line 5, LB-RLT evaluates the performance score of the current node.

Algorithm 1: LB-RLT PERFORMANCE SCORE EVALUATION

```
1 Input: replicasNodeList
2 Output: replicasNodeList
3 foreach node in the replicasNodeList do
4     node.performanceScore ← node.lastReadDataChunkThroughput ×
5     node.totalReadDataChunks
6 return replicasNodeList
```

Analyzing the Algorithm 2, on line 1, the input objectData is the data object to be read by the proxy node from a replica node. On line 2, the input replicasNodeList is all the replica nodes that can be used to retrieve the data object. On line 3, the output dataChunk[] is the data chunks reads of the object data. On line 4, the Algorithm 1 is performed to evaluate the performance score of all the replica nodes. On line 5, the list of storage nodes is sorted according to our load balancing approach. In case of similar performance score among the replica nodes, the replicaNodeList is sorted using round-robin. On line 6, the replica node with best performance score is chosen to start the reading of the data object. On line 7, proxy node begins to read all the data object dataChunk by dataChunk from the storage node. On line 9, it is verified if the reading of the dataChunk has succeeded. In positive case, on line 10 and on line 11, for the current best node, the lastDataChunkThroughput and the totalReadDataChunks metrics are updated.

Algorithm 2: LB-RLT - OBJECT READ REQUEST

```
1  Input: objectData
2  Input: replicasNodeList
3  Output: dataChunk[]
4  replicasNodeList ← evaluateLBRLTPerformanceScore(replicasNodeList)
5  replicasNodeList ← sort(replicasNodeList)
6  bestNode ← getFirstNode(ReplicasNodeList)
7  foreach dataChunkIter in the objectData do
8      dataChunk[i] ← readDataChunkFrom(bestNode)
9      if ReadDataChunkSucceeded then
10         bestNode.lastDataChunkThroughput ← dataChunkSize /dataChunkTransferTime
11         bestNode.totalReadDataChunks += 1
12 return dataChunk[]
```

3. Environment

There are many cloud object storage solutions available, such as Ceph, GlusterFS and OpenStack-Swift. In this work, we used OpenStack-Swift that is a highly available, distributed, eventually consistent object/blob store [Swift 2016]. For evaluating our experiment, we chose Swift due its simplicity, once it seemed easier to implement our load balancing approach. We use the Ubuntu 12.04 operating system and Swift Kilo version.

3.1. OpenStack-Swift

Swift architecture can hold many different web service nodes and background processes whereas each component can be scaled multiple times. The web service nodes are classified in two main categories: proxy server type and storage node type (composed by account server, container server and object server). The background processes are responsible for data replication, data reconstruction, data updating and data auditing. The proxy server node takes requests from a client and forwards them to the account, container and object server nodes in order to persist/retrieve data objects and its metadata to/from disks. Swift implements a ring (consistent hashing) in order to map data and its replicas from the logical locations to their physical locations.

Swift implements partial replication model and flexible configuration for number of nodes and number of replicas. Read and write requests from/to Swift obeys the following rules: for reading requests, it is enough only one replica node to retrieve the object and return success to the client; for writing requests, it is enough the quorum of half of total replica nodes plus one to save the object and return success to the client.

3.2. Benchmark

In order to evaluate our approach, we used *Cloud Object Storage Benchmark* (COSBench) [Zheng et al. 2013]. COSBench has support to many different cloud object storage solutions on the market like Swift, Amazon S3, and Ceph thus making easier any future comparison among those cloud object storage solutions. We use the COSBench 0.4.0.1 version.

For our workload, we fixed read/write ratio at 80/20. We varied tiny sizes for object data size: 4KB and 16KB. For each object data size, we varied the number of threads/workers: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 and 700 in order to compare the approaches in different scales. We did not vary the delay between requests. The performance of the load balancing approaches was compared using the average values of the benchmark metrics collected in a interval of 5 seconds by COSBench with transaction phase of 300s and the default values for other COSbench parameters. In this paper, we considered three metrics for evaluating the approaches: average response time, throughput and success ratio.

3.3. Hardware

For our hardware configuration, we deployed five different physical machine nodes. One node for COSBench and four nodes for a minimalist Swift deployment. For our Swift deployment, we configured one proxy node with more processing power and three heterogeneous nodes. Storage Node 02 (SN2) and Storage Node 03 (SN3) presented best performance, whereas Storage Node 01 (SN1) presented poor performance during our initial performance profiling. Hardware details are described at the Tables 3 and 4.

Storage Node (SN1) was chosen on purpose, with a disk architecture that offers less IO performance than we, as Table 4. This aspect tries to play a scenario with heterogeneous nodes in a data center.

For our swift deployment with three storage nodes, we configured three replicas for each data object just because it is a common setup and it does not make sense to have more replicas than the number of nodes even though our approach works is agnostic to the number of replicas and nodes.

Table 3. CPU Configuration

Node	Cores	MHz	L1 / L2 / L3 (Cache)
COSBench	8	800	32 / 256 / 8192
Swift Proxy	8	800	32 / 256 / 8192
SN1	2	**1995**	32 / **2048** / N/A
SN2	4	1600	32 / 256 / 6144
SN3	8	800	32 / 256 / 8192

Table 4. RAM and Disk Configuration

Node	RAM(GB)	R/RW(IOPS)	Filesystem
COSBench	8	133 / 100,4	xfs
Proxy Node	16	161,4 - 123,8	xfs
SN1	3	**98,4 / 80,4**	xfs
SN2	8	147,8 / 119,4	xfs
SN3	4	142,6 / 121,4	xfs

IOPS benchmark performed by random disk operations for a workload of 4KB by 16 threads simultaneously using FIO - Flexible I/O Tester Synthetic Benchmark.

4. Evaluation

In this section, we describe the results we have obtained from it. According to the available Swift configuration, we verified three different load balancing approaches implemented by the Swift object storage, thus making our approach comparable against production-like approaches described in the Table 5.

Approaches under evaluation are classified according to their types and needed for profiling as shown in the Table 6. For LB-RLT evaluation against other approaches, we focused on analyzing which load balance approach obtained best results from the premise described in COSBench paper which considers a better system one that is able to engage more clients with same resources [Zheng et al. 2013]. In our experiments, we configured the LB-RW approach to give preference to the best two nodes (SN2 and SN3).

4.1. Replica Selection Monitoring

The replica selection behavior of the load balancing approaches was monitored enabling ”statsd” time-series metrics [Malpass and Malpass 2011] via UDP protocol for every Swift node in order to confirm which storage nodes were being chosen for every load balancing approach. Below, we describe what happened in the experiment for 16KB as object size since the overall behavior of the approaches were similar for 4KB as object size.

Table 5. Approaches performed

Notation	Description
LB-R	Load Balance by **Round-robin** distribution. In the Openstack Swift, this is called the shuffling method (tested) [Ying et al. 2015].
LB-RW	Load Balance by Round-robin with static **Weights**. Profiling of nodes performance is needed. In the Openstack Swift, this is called the affinity method (tested).
LB-RCT	Load Balance by Round-robin and **Connection Timing**. In the Openstack Swift, this is called the timing method (tested).
LB-RL	Load Balance by Round-robin and **LastReadDataChunkThroughput** performed by each storage node (proposed and tested).
LB-RLT	Load Balance by Round-robin, **LastReadDataChunkThroughput** and **TotalReadDataChunks** performed by each storage node, i.e., total of dataChunks successfully read (proposed, tested and the best choice).

Table 6. Compared Approaches

Approach	Profiling	Run-time measurement	distribution criteria	Standard
LB-R	N/A	No	Round-robin	Yes
LB-RW	Needed	No	Weights	Yes
LB-RCT	N/A	Yes	Response-time	Yes
LB-RL	N/A	Yes	Throughput	Custom
LB-RLT	N/A	Yes	Throughput and Total succeeded Reads	Custom

The LB-R approach balanced equally the workload for all the three storage nodes. Since it is the obvious load balancing behavior, we did not show its monitoring in the Figure 2. LB-RCT had similar results in comparing with LB-R, but it got worse even though LB-RCT also used round-robin technique. In the beginning of the experiment, LB-RCT gave preference to the worse node (SN1). As LB-RCT strategy uses no disk access, since it performs only an HTTP connection test, this happened mainly because SN1 has the largest L2 memory cache. This can be verified at the Item C of the Figure 2, for the LB-RTC approach, the continuous line (SN1) stayed above the dashed lines (SN2 and SN3). According to [Foong et al. 2003], when transfer sizes are less than L2, the cache is large enough to accommodate all of the application buffers. In LB-RCT case, L2 cache memory made TCP connection faster for the worse storage node (SN1).

LB-RW balanced the workload only to the SN2 and SN3 according to the previously configured weights with exception when the benchmark tool began to overload the cloud object storage in such a way that was impossible its proper usage. By this means, Swift implementation tries any available replica node. So, it is clear, at the Figure 2, that LB-RW used SN1 (worse storage node) only when the system was already unable to respond properly the requests.

LB-RL balanced almost equally the workload to all storage nodes. We believe this happened because the way we implemented LB-RL caused the replica selection to prefer nodes with higher throughput values without considering which nodes were processing more successful readings, thus not being enough to evaluate storage node performance dynamically.

Our approach LB-RLT load balanced the workload with three behaviors: (1) at the very beginning, SN2 or SN3 was used because only one was able to respond the requests,

(2) after this very beginning and before the half of the experiment, both SN2 and SN3 were used and (3) before a little bit the half of the experiment, both SN2 and SN3 continued to be used, but LB-RLT started to use the SN1. This last behavior happened because the two best nodes were not enough for processing the increased workload as described by the continuous line representing the SN1 at the Figure 2.

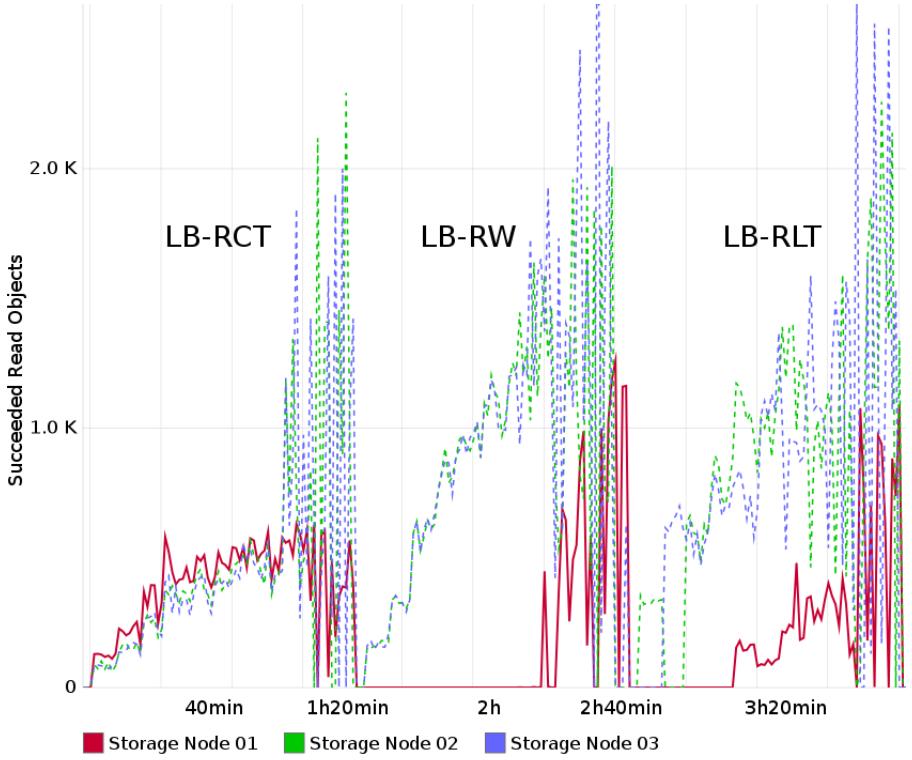


Figure 2. Load Balancing Monitoring for 16KB.

4.2. Performance Comparison

LB-RLT and LB-RW performed more operations with success than the other approaches. This average of total of operations performed with success is shown at the Table 7.

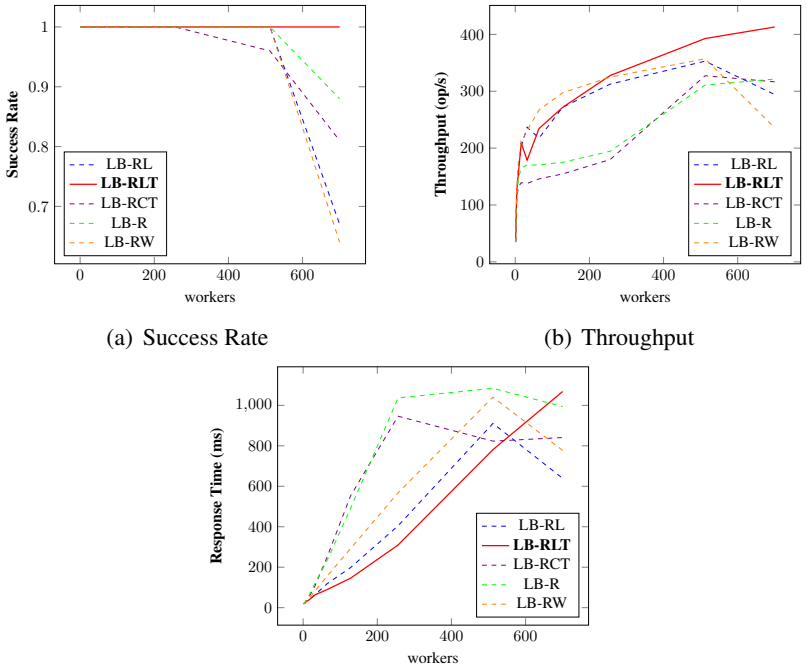
Table 7. Number of operations (avg. ops) for 100% Success Rate

Approach	4KB read	16KB read	4KB write	16KB write
LB-R	1.66×10^4	1.10×10^4	3.70×10^3	2.10×10^3
LB-W	1.90×10^4	1.44×10^4	4.46×10^3	2.85×10^3
LB-RCT	1.50×10^4	1.09×10^4	3.16×10^3	2.06×10^3
LB-RL	1.85×10^4	1.08×10^4	4.30×10^3	1.99×10^3
LB-RLT	1.96×10^4	1.45×10^4	4.65×10^3	2.87×10^3

Comparing the success rate metric with 4KB as the object size, only LB-RLT performed all the read requests with 100% success rate as shown by the continuous line at the Figure 3(a). A similar behavior happened with 16KB as the object size because LB-RLT lasted longer in 80% of success rate as shown by the continuous line in the Figure 4(a). LB-RCT and LB-R were totally worse than the others. Our environment was deployed to not support purposely a high number of workers with 16KB as object size (above cloud capacity scenario).

Comparing the throughput metric with 4KB as the object size, LB-RW could not maintain the highest throughput after 200 workers. LB-RL had similar results with LB-RW. From the middle to the end, LB-RLT bypassed LB-RW as shown by the continuous line in the Figure 3(b). In the comparison with 16KB as the object size, LB-RW could not maintain again the best throughput after 200 workers as shown in the Figure 4(b). Considering the lower success rates during the rest of the experiment, LB-RLT maintained a little bit better throughput than all the other approaches. LB-RL had a bad throughput similar to LB-RCT and LB-R because LB-RL approach could not give preference to the best nodes as discussed at Section 4.1.

Comparing the response time metric with 4KB as the object size, LB-RLT performed the lowest response time during all experiment until in the end when the success rate 4(a) of the other approaches began to fall as shown at the Figure 3(c). The comparison with 16KB as object size, LB-RLT had the best response time until near after success rate began to fall from 100% 4(a) as shown in the Figure 4(c). The response time metric for LB-RLT increased in the end because LB-RLT processed more requests than the other approaches, once the others were losing requests. LB-RL again did not perform well for the same reason described at the throughput comparison. The response time metric is available and is statistically analyzed at the Section 4.3



(a) Success Rate

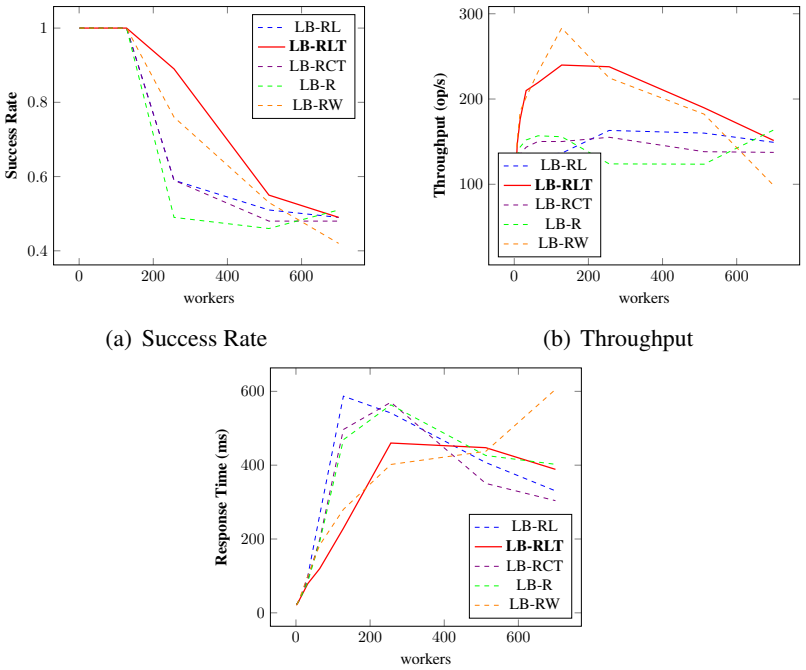
(b) Throughput

(c) Response Time

Figure 3. Load Balance Approaches Comparison For 4KB object sizes

4.3. Response Time Analysis

For 4KB as object size, at Table 8, LB-RLT performed on average a response time 11.58% better than any other approach. In the best case, the response time was 36.11% better and in the worst case only 1,93% worse. The experiment with 256, 512 and 700 workers were not considered because not all the approaches have completed 100% of successful operations.



(a) Success Rate (b) Throughput

(c) Response Time

Figure 4. Load Balance Approaches Comparison For 16KB object sizes

Table 8. Response time analyses for 4KB object size

Workers	**LB-RTL**	LB-RL	LB-RCT	LB-R	LB-RW	Best others	(Best others/LB-RLT) - 1
1	20.53	20.97	20.78	20.72	20.88	20.72	.93%
2	20.33	20.08	21	20.09	20.22	20.08	-1.23%
4	21.71	21.29	23.01	25.38	21.5	21.29	-1.93%
8	28.62	32.67	32.45	36.48	31.08	31.08	8.6%
16	38.74	40.36	53.99	61.74	47.06	40.36	4.18%
32	63.46	64.04	108.48	120.28	82.83	64.04	.91%
64	88.81	117.44	247.5	231.19	153.72	117.44	32.24%
128	145.79	198.43	553.5	491.46	292.49	198.43	36.11%
256	309.73	404.59	946.49	1036.82	567.81	404.59	30.63%
512	781.11	910.85	823.13	1084.23	1040.42	823.13	5.38%
						mean	11,58%
						best	36,11%
						worse	-1,93%

For 16KB as object size, at Table 9, LB-RLT performed on average a response time 10.89% better than any other approach. In the best case, the response time was 53.46% better and in the worst case 3.57% worse. The experiment with 256, 512 and 700 workers were not considered because none approach has completed 100% of successful operations.

Table 9. Response time analyses for 16KB object size

Workers	**LB-RLT**	LB-RL	LB-RCT	LB-R	LB-RW	Best others	(Best others/LB-RLT) - 1
1	22.19	23.1	22.68	22.78	22.96	22.68	2.21%
2	21.95	23.93	23.52	23.46	22.93	22.93	4.46%
4	25.47	26.48	25.21	25.01	24.56	24.56	-3.57%
8	32.6	32.36	32.82	32.9	35.82	32.36	-.74%
16	47.99	49.41	50.52	48.92	55.84	48.92	1.94%
32	79.16	100.09	90.82	84.67	100.54	84.67	6.96%
64	119.63	264.74	198.83	192.32	183.59	183.59	53.46%
128	229	586.96	496.63	468.89	280.23	280.23	22.37%
						mean	10.89%
						best	53.46%
						worse	-3.57%

5. Related Work

LB-R and LB-RW can be classified as static approaches according to [Khiyaita et al. 2012] and [Katyal and Mishra 2013]. LB-R approach is not able to adapt to any change during runtime, thus making it a bad choice for heterogeneous storage nodes environment. LB-RW approach allows finer control according to prior knowledge of nodes' capacity. It basically sets a weight for each node. In case of same weights for storage nodes, equally sorting by round-robin is used to spread the workload. A drawback for LB-RW is mainly due to its static behavior for high loads, once the approach does not collect any metric of the system current performance.

LB-RCT is mentioned as Central Load Balancing Decision Model (CLBDM) by [Katyal and Mishra 2013] [Radojević and Žagar 2011] to be used in a static environment, but it uses a runtime metric to load balance the nodes. Basically, LB-RCT measures the duration required to establish a TCP connection between a proxy node and a storage node in order to give preference to the storage nodes with lower connection latency. It also implements measure expiration protection and equally sorts nodes in case of same latency environment. A drawback for this approach is due to the fact that latency of connection does not consider other system aspects such as disk performance.

LB-RL approach is cited by [Katyal and Mishra 2013] as a throughput issue to be addressed for static, dynamic, centralized and distributed types of algorithms. We analyzed our LB-RL implementation on OpenStack-Swift as a dynamic load balance approach by using the lastReadDataChunkThroughput measured between the proxy node and the storage nodes with measure expiration protection and equally-sorting nodes in case of same throughput. A drawback for this approach is due to the fact that, according to our experiments, using only the lastReadDataChunkThroughput is not enough because the bad node continued to be selected for reading more than it was needed.

Our approach aimed to bypass limitations from the previous approaches by using the lastReadDataChunkThroughput that considers a last storage node performance and the

totalReadDataChunks as a long-term performance of storage node until present moment. By this means, in runtime, LB-RLT is able to give preference to the best nodes without neglecting the bad nodes when needed.

6. Conclusion and Future Work

This work presented LB-RLT, an approach for load balancing replica selection in a cloud object storage heterogeneous system. LB-RLT collects two metrics: lastReadDataChunk-Throughput and totalReadDataChunks, and combines both metrics in order to adapt not only for storage nodes heterogeneous environments but also for when load requirement changes dynamically. We evaluated the LB-RLT approach by considering different performance metrics, obtaining the lowest response time, the highest throughput rate and lasting success rate, without any prior performance profiling. Considering our results, LB-RLT improves the overall performance of cloud object storage systems.

As future work, we intend to conduct new experiments with different proportion between read and write requests as well as more scenarios for workloads. Other important issues to be addressed are related to experiments with a greater number of nodes and the analysis of expiration time when there is more delay between the coming requests.

Acknowledgments

This work is partially supported by Hitachi Data Systems (HDS) and Laboratório de Sistemas e Banco de Dados (LSBD) - UFC.

References

Allcock, B., Bester, J., Bresnahan, J., Chervenak, A. L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., and Tuecke, S. (2002). Data management and transfer in high-performance computational grid environments. *Parallel Computing*, 28(5):749–771.

Brinkmann, A., Salzwedel, K., and Scheideler, C. (2000). Efficient, distributed data placement strategies for storage area networks. In *Proceedings of the twelfth annual ACM symposium on Parallel algorithms and architectures*, pages 119–128. ACM.

Foong, A. P., Huff, T. R., Hum, H. H., Patwardhan, J. P., and Regnier, G. J. (2003). Tcp performance re-visited. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 70–79. IEEE.

Godfrey, B., Lakshminarayanan, K., Surana, S., Karp, R., and Stoica, I. (2004). Load balancing in dynamic structured p2p systems. In *INFOCOM 2004*, volume 4, pages 2253–2262. IEEE.

Honicky, R. and Miller, E. L. (2004). Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 96. IEEE.

Katyal, M. and Mishra, A. (2013). A comparative study of load balancing algorithms in cloud computing environment. *International Journal of Distributed and Cloud Computing*, 1(2).

Khiyaita, A., Zbakh, M., El Bakkali, H., and El Kettani, D. (2012). Load balancing cloud computing: State of art. In *Network Security and Systems (JNS2), 2012 National Days of*, pages 106–109.

Lumb, C. R., Ganger, G. R., and Golding, R. (2003). D-sptf: Decentralized request distribution in brick-based storage (cmu-cs-03-202).

Malpass, I. and Malpass, I. (2011). Measure anything, measure everything.

Mesnier, M., Ganger, G. R., and Riedel, E. (2003). Object-based storage. *Communications Magazine, IEEE*, 41(8):84–90.

Radojević, B. and Žagar, M. (2011). Analysis of issues with load balancing algorithms in hosted (cloud) environments. In *MIPRO, 2011 Proceedings of the 34th International Convention*, pages 416–420. IEEE.

Rajalakshmi, A., Vijayakumar, D., and Srinivasagan, K. (2014). An improved dynamic data replica selection and placement in cloud. In *Recent Trends in Information Technology (ICRTIT), 2014 International Conference on*, pages 1–6. IEEE.

Swift (2016). *OpenStack*. `http://docs.openstack.org/developer/swift/`.

Weil, S. A., Brandt, S. A., Miller, E. L., and Maltzahn, C. (2006). Crush: Controlled, scalable, decentralized placement of replicated data. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 122. ACM.

Xie, Q., Dong, X., Lu, Y., and Srikant, R. (2015). Power of d choices for large-scale bin packing: a loss model. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 321–334. ACM.

Ying, L., Srikant, R., and Kang, X. (2015). The power of slightly more than one sample in randomized load balancing. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 1131–1139. IEEE.

Zheng, Q., Chen, H., Wang, Y., Zhang, J., and Duan, J. (2013). Cosbench: cloud object storage benchmark. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pages 199–210. ACM.

Trilha Principal do SBRC 2016
Sessão Técnica 8
Redes Ópticas

Uso de Aspectos da Topologia Virtual no Problema RWBA em Redes Ópticas Metropolitanas MB-OFDM

Eduardo S. Gama[1], Iguatemi E. Fonseca[1], Carlos M. Araújo[1], Raul C. Almeida Júnior[2], Tiago M. F. Alves[3], João P. F. Rosário[3], Adolfo V. T. Cartaxo[3]

[1]Centro de Informática – Universidade Federal da Paraíba (UFPB)
João Pessoa - PB, Brasil

[2]Departamento de Eletrônica e Sistemas – Universidade Federal de Pernambuco (UFPE)
Recife - PE, Brasil

[3]Instituto de Telecomunicações IST – Universidade de Lisboa
1049-001 Lisboa, Portugal

iguatemi@ci.ufpb.br, raul.almeidajunior@ufpe.br, adolfo.cartaxo@lx.it.pt

Abstract. *In this paper the RWBA problem applied to virtual/logical topology of multi-band optical orthogonal-frequency-division-multiplexing (MB-OFDM) metropolitan (metro) networks is analysed. A MORFEUS metropolitan network consists of a ring topology network with bidirectional links. Given a connection request, the RWBA algorithm must assign a route, a wavelength and a set of bands in order to serve the request. In the MORFEUS network architecture, each ROADM (Reconfigurable Optical Add-Drop Multiplexer) node is composed by WDM multiplexing components, and components for insertion and extraction of MB-OFDM bands. These last components are called respectively MIB (MORFEUS Insertion Block) e MEB (MORFEUS Extraction Block). In a MORFEUS node, a WDM MB-OFDM signal can have bands extracted or inserted, or even pass through with no alterations. In order to optimize the MIB and MEB usage, was proposed in this paper the use of RWBA algorithm applied to the logical topology of the network. Numerical simulations suggest that a redution of 50% in the number of wavelengths and MIB/MEB's may be achieved when the MORFEUS network operates with a RWBA algorithm that taken into account aspects of the virtual topology in optical networks. Moreover, when the network capacity is increased, the blocking performance of the RWBA with and without aspects of the virtual topology are similar.*

Resumo. *Este artigo trata do uso de aspectos do projeto de topologia virtual no problema de Alocação de Rota, Comprimento de Onda e Banda (RWBA - Routing, Wavelength and Band Assignment) em redes ópticas metropolitanas MB-OFDM. A rede metropolitana MORFEUS considerada neste trabalho é uma rede de topologia em anel com enlaces bidirecionais. Dada uma requisição de conexão, o algoritmo RWBA deve atribuir uma rota, um comprimento de onda e um conjunto de bandas capazes de atendê-la. Na arquitetura da rede MORFEUS, cada nó ROADM (Reconfigurable Optical Add-Drop Multiplexer) é composto por componentes de multiplexação WDM e componentes de extração e inserção de banda MB-OFDM, chamados, respectivamente, de MEB (MORFEUS Extraction Block) e MIB (MORFEUS Insertion Block). Com isso um*

sinal WDM MB-OFDM pode ter banda(s) extraída(s) e/ou inserida(s) num nó MORFEUS, ou ser direcionado para a fibra de saída sem alteração. Com o intuito de otimizar o uso dos componentes MIB/MEB's, bem como o número total de comprimentos de onda na rede, este artigo propõe uso de aspectos do projeto de topologia virtual em redes ópticas no problema RWBA. Os resultados das simulações numéricas mostram que uma economia de até 50% no número de comprimentos de onda e MIB/MEB's é alcançada quando comparada com o mesmo algoritmo RWBA que não leva em consideração os aspectos da topologia virtual. Além disso, quando a capacidade da rede MB-OFDM é incrementada, os RWBA's nos cenários com e sem aplicação da topologia virtual apresentam probabilidade de bloqueio equivalentes.

1. Introdução

A abordagem MB-OFDM (*Multi-Band Optical Orthogonal Frequency Division Multiplexing*) usando DD (*Direct Detection*) foi proposta para redes metropolitanas ópticas de grandes distâncias [Alves et al. 2015], [Blouza et al. 2011], [Peng et al. 2012], [Peng et al. 2009], [Li et al. 2013], [Alves et al. 2014] e [Cartaxo et al. 2014]. No cenário de redes metropolitanas flexíveis, o uso de uma portadora para cada banda OFDM foi proposta para auxiliar a detecção das bandas e reduzir a largura de banda necessária nos receptores *front-end*.

Uma rede metropolitana MB-OFDM empregando portadoras virtuais com detecção direta, aqui chamada de MORFEUS, foi proposta em [Alves et al. 2014], e trabalhos recentes discutiram sua arquitetura, limites de transmissão, bem como o desempenho da sua taxa de erro de bit [Alves et al. 2015], [Alves et al. 2014], e [Mendes et al. 2014]. A rede MORFEUS é baseada no uso de WDM (*Wavelength Division Multiplexing*) de forma que cada comprimento de onda (canal óptico) pode ser retido ou passar adiante em cada nó. Quando o sinal MB-OFDM chega a um dado nó da rede, um filtro óptico convencional é usado para selecionar a banda (junto com a portadora virtual) e extraí-la. Usando um conjunto de bandas em cada canal óptico, e assumindo que cada cliente pode requisitar uma dada quantidade de bandas, temos um novo problema de atribuição de rota, comprimento de onda e conjunto de bandas (RWBA - *Routing, Wavelength and Band Assignment*). Em [Fonseca et al. 2015], novos algoritmos para o problema RWBA foram propostos e estudados, bem como a relação entre o desempenho das probabilidades de bloqueio e as configurações do espectro (quantidade de comprimentos de onda e bandas OFDM).

Em uma rede do tipo MORFEUS, cada nó é composto por componentes de multiplexação usados para filtrar as portadoras geradas por WDM, e também componentes ópticos de extração e inserção de bandas OFDM de forma totalmente óptica. Sinais WDM MB-OFDM que chegam a um nó, primeiro passam por um demultiplexador e, em seguida, são direcionados para os blocos de extração MEB's (*MORFEUS Extraction Block*) e/ou inserção MIB's (*MORFEUS Insertion Block*) através *switches* ópticos. Os MIB's e MEB's são responsáveis pela inserção e extração de bandas para atender o tráfego dos usuários, respectivamente. Levando em consideração o custo de produção dos MEB's e MIB's, nesse artigo é estudada uma estratégia de aplicação do algoritmo RWBA que leva em conta aspectos da topologia virtual da rede óptica como forma de

otimizar o uso dos MEB's e MIB's, diminuindo a quantidade necessária desses componentes em cada nó MORFEUS a fim de baratear o custo de implantação da rede. Os resultados das simulações numéricas mostram que uma economia de até 50% no número de comprimentos de onda e MIB/MEB's é alcançada quando comparada com o mesmo algoritmo RWBA que não leva em consideração os aspectos da topologia virtual. Além disso, quando a capacidade da rede MB-OFDM é incrementada, os RWBA's nos cenários com e sem aplicação da topologia virtual apresentam probabilidade de bloqueio equivalentes. Para o melhor do nosso conhecimento, essa é a primeira vez na literatura que algoritmos RWBA são investigados levando-se em consideração aspectos da topologia virtual em redes ópticas.

O restante deste artigo está organizado da seguinte maneira. Na Seção 2 é apresentada a arquitetura dos nós da rede MORFEUS, bem como aspectos funcionais dos MIB/MEB's. A Seçao 3 apresenta como a estratégia de uso da topologia virtual é incorporada aos algoritmos RWBA em uma rede WDM MB-OFDM. A Seção 4 mostra e debate detalhadamente os resultados das simulações numéricas sob o ponto de vista de diferentes métricas de desempenho. Finalmente, na Seção 5 as conclusões são sumarizadas e trabalhos futuros são apontados.

2. Nó MORFEUS

Na Figura 1 é ilustrada a arquitetura esquemática do nó MORFEUS. Como já mencionado, cada nó MORFEUS é composto essencialmente por componentes de multiplexação que tratam sinais WDM e MB-OFDM, nessa ordem.

O sinal WDM MB-OFDM que chega ao nó passa primeiro por um demultiplexador WDM que detecta o comprimento de onda da portadora. As saídas do demultiplexador estão conectadas à *switches* ópticos que podem encaminhar o sinal, agora MB-OFDM, para os componentes de extração e inserção (MEB's e MIB's) ou para o multiplexador que está ligado a fibra de saída do nó. O trio demultiplexador - *switch* óptico - multiplexador compõe o ROADM (*Reconfigurable Optical Add-Drop Multiplexer*).

Quando o *switch* óptico está transversalmente posicionado, o sinal MB-OFDM é encaminhado para os blocos de extração (MEB's). Os MEB's apresentam duas saídas: uma ligada aos blocos de inserção (MIB's) e outra ligada a um segundo *switch* óptico responsável por lançar esse sinal em uma outra rede metropolitana ou em uma rede de acesso conectada ao nó. Os MIB's permitem inserir novas bandas OFDM a *slots* livres no sinal MB-OFDM.

Quando o *switch* óptico estiver na posição paralela, o sinal MB-OFDM é encaminhada para o multiplexador sem nenhuma alteração na extração ou inserção da banda. Ou seja, o sinal WDM MB-OFDM passa de um nó para outro na rede de forma transparente.

Como mencionado anteriormente, o uso do sinal WDM MB-OFDM introduz o problema de escolha adequada da rota, comprimento de onda e um subconjunto de bandas (RWBA) a serem usados pela requisição. Uma vez que diferentes granularidades (i.e., comprimento do canal óptico e, consequentemente número de bandas por canais ópticos) possam ser adotados e utilizados na rede MORFEUS, este artigo também investiga a importância de reaproveitar os caminhos ópticos existentes na rede, por meio do uso de conceitos de topologia lógica/virtual em redes ópticas. Na próxima seção tais aspectos da topologia lógica são apresentados e debatidos.

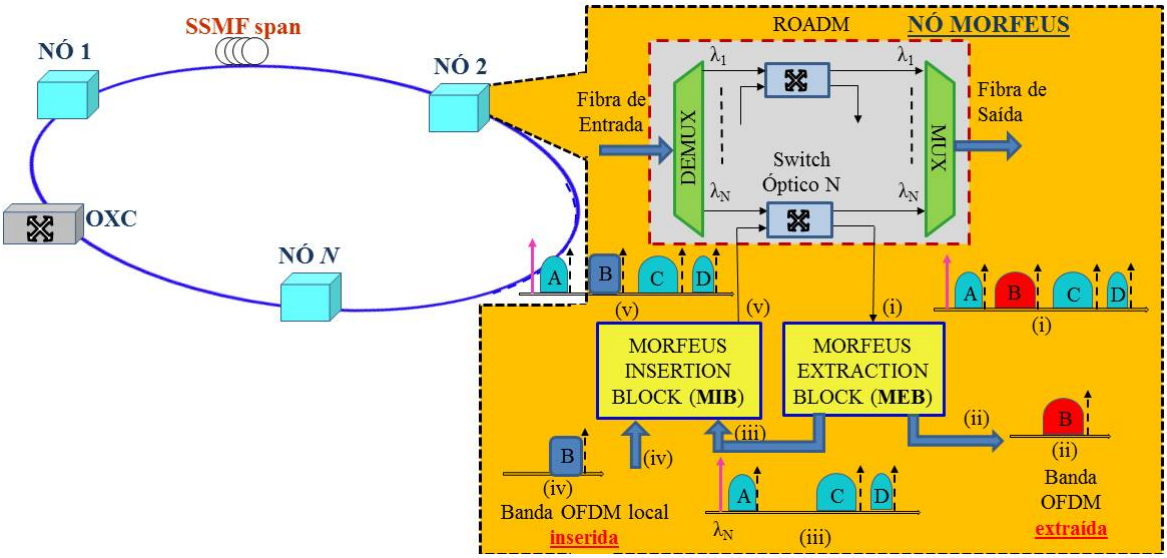


Figura 1. Arquitetura esquemática do nó MORFEUS.

3. Topologia Lógica na rede MORFEUS

Apesar do componente ROADM, responsável por encaminhar sinais WDM que não apresentam necessidade de alteração (extração ou inserção), o nó MORFEUS não é capaz de tratar de forma eficiente a alteração parcial do sinal WDM. Isso significa que para alterar parcialmente um sinal, todo ele é analisado, ou seja, parar extrair parte das bandas do sinal, como é mostrado na Figura 1 (ii), todas as bandas precisam ser tratadas por MEB's. Os sinais de banda OFDM resultantes da extração são então encaminhados de acordo com as conexões relacionadas. Na Figura 1 após a extração das bandas, a banda B é encaminhada para uma rede externa enquanto o resto das bandas são encaminhadas para os MIB's, onde uma nova banda B pode ou não ser inserida (ver (iv)). Entretanto, para isso, foi necessário a extração e reinserção das bandas A, C e D.

Visando evitar esse tipo de situação de reinserção de bandas e ativação de novos comprimentos de onda na rede, este trabalho propõe o uso de topologias lógicas capaz de transportar as bandas de uma requisição fim-a-fim minimizando o uso de blocos de extração e inserção de bandas da rede, bem como o número de comprimentos de onda em uso na rede. A topologia lógica oferece um certo grau de independência em relação à topologia física, diferentes topologias lógicas podem ser configuradas na mesma topologia física, embora o conjunto de todas as topologias lógicas que podem ser criadas é limitada pela topologia física. Nesse trabalho, uma nova abordagem do algoritmo RWBA é apresentada, a qual usa a topologia lógica de conexões estabelecidas como base prioritária para busca e alocação de comprimento de onda e banda.

3.1. Algoritmo da Topologia Lógica

A Figura 2 apresenta o problema RWBA [Fonseca et al. 2015] com o uso da Topologia Lógica (TL) proposto neste trabalho. O uso da TL em RWBA, primeiro, procura rotas nos grafos G_{tl}^{w} com o intuito de reaproveitar as conexões existentes na rede, alocando as bandas da requisição gerada em comprimentos de onda que estão em uso. Como é assumido que, inicialmente, a rede não possui nenhuma conexão, assim, $G_{tl} = NULL$, os

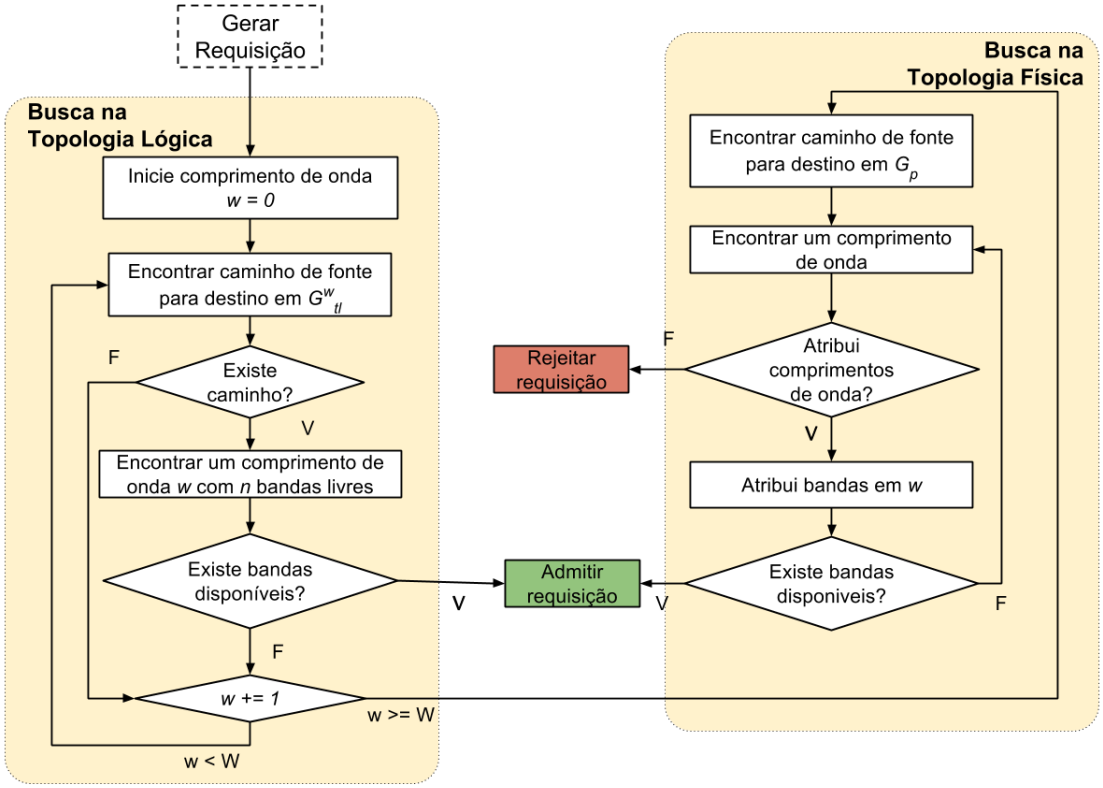


Figura 2. Fluxograma do problema RWBA com o uso da Topologia Lógica.

grafos de cada comprimento de onda w são preenchidos a medida em que os subproblemas de atribuição de comprimento de onda (WA - *wavelength assignment*) e atribuição de bandas (BA - *band assignment*) aloca as requisições na rede. Como pode-se ver na Figura 2, quando um caminho é encontrado com o uso da TL, o algoritmo executa apenas o subproblema BA com o comprimento de onda w que contenham requisições ativas. Uma informação importante a ser percebida no algoritmo RWBA com topologia lógica é que as requisições nunca são bloqueadas quando a busca na TL é executada, elas tem o objetivo de reaproveitar as conexões já criadas, adicionando um caminho de ***origem*** para ***destino*** em G^w_{tl}.

Para adequar o problema RWBA com a TL foi feita uma adaptação no subproblema WA. O algoritmo busca primeiro os W comprimentos de onda totalmente livres no caminho encontrado no subproblema de atribuição de rota (RA - *routing assigment*), com um conjunto contiguo de bandas livres, capaz de atender a requisição. Após os W comprimentos de onda serem encontrados, a alocação ocorre de acordo com a escolha das heurísticas implementadas nos subproblemas WA e BA, respectivamente.

A matriz de preenchimento da topologia lógica é feita no momento da alocação e desolacação das requisições na rede. No momento da alocação de uma requisição do nó fonte ao nó destino e um comprimento de onda w estiver totalmente desocupado, um novo enlace em G^w_{tl} é criado, caso contrario, a matriz de preenchimento não altera o estado, permanecendo o mesmo. No momento da desalocação de uma requisição na rede a matriz de preenchimento pode passar por uma modificação em G^w_{tl}, se a requisição a ser

liberada no comprimento de onda w pelo caminho do nó fonte ao nó destino não conter mais nenhuma requisição o enlace será removido, caso contrário, o enlace permanece.

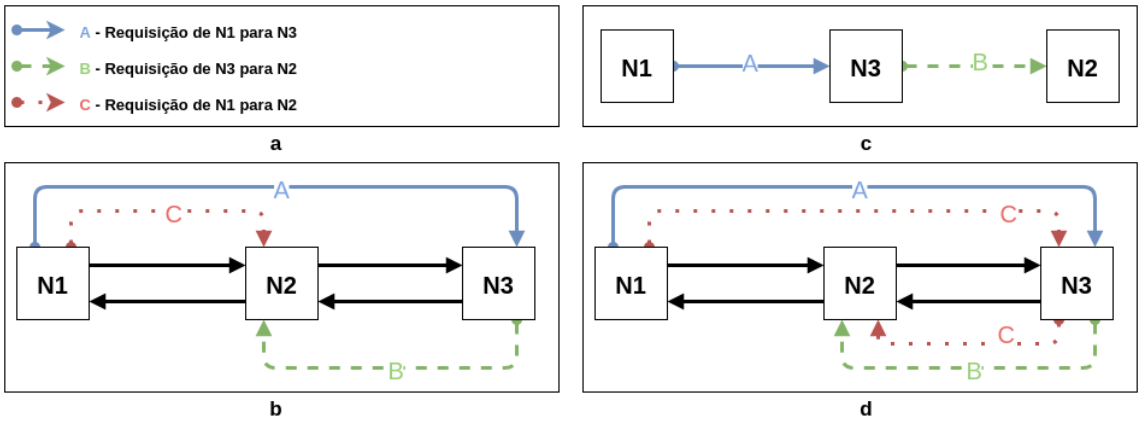


Figura 3. (a) legenda de requisições, (b) alocação de rota baseada na topologia física, (c) topologia lógica de conexões estabelecidas pelas requisições A e B, (d) alocação de rota baseada na topologia lógica.

A Figura 3 ilustra como um conjunto de requisições (a) podem ser alocadas de formas diferentes em relação as rotas escolhidas. Em (b) a rota da requisição C é alocada baseada na topologia física e em (d) é alocada baseada na topologia lógica (c). Dadas as requisições A e B mostradas na Figura 3(a), uma terceira requisição C pode ser alocada de forma eficiente evitando a extração e reinserção da informação da requisição A no nó N2, como é mostrado em (b). Baseando-se na topologia lógica mostrada em (c), em que cada *lightpath* estabelece uma aresta entre seus nós origem e destino, uma nova alocação de rota pode ser estabelecida, mostrada em (d), evitando que, em N2, o sinal da requisição A seja tratado e reinserido. Por consequência, no nó N3 o sinal da requisição C é extraído e inserido em um novo *lightpath* junto com o sinal da requisição B. Nesse último caso, o uso dos MIB's e MEB's não é considerado ineficiente, uma vez que a extração e inserção nesse ponto é inevitavelmente necessária.

Dessa forma, o uso da topologia lógica no problema RWBA visando o reaproveitamento de *lightpaths* tem o objetivo de evitar dois tipos de situação. Dado um determinado comprimento de onda, deve-se evitar que *lightpaths* usem como nós intermediários aqueles que são: (a) apenas origem (apliquem apenas inserção de bandas) ou (b) apenas destino (apliquem apenas extração de bandas). Casos onde os nós intermediários são tanto origem como destino são aceitáveis. Vale salientar que, devido a restrição de continuidade, é necessário uma topologia lógica independente para cada comprimento de onda definido no espectro.

Este trabalho considera uma topologia em anel para a rede MORFEUS. Para o subproblema de roteamento (RA - *Routing Assignment*) é usado algoritmo de Dijkstra para a escolha das rotas de menor caminho considerando cada salto para os nós vizinhos com peso igual a 1, assim, nos concentramos nos subproblemas WA e BA. Um comprimento de onda é dito disponível se apresentar o mesmo subconjunto de um subconjunto contíguos de *slots* livres entre os enlaces da rota escolhida para serem alocados.

Neste trabalho foi investigada a heurística *First-Fit* (FF), na qual o primeiro comprimento de onda livre ou subconjunto de bandas livres, respectivamente, no problema

WA ou BA, de uma preestabelecida ordem é selecionada. O FF é utilizado em dois cenários de rede diferentes. O primeiro é idêntico ao apresendo em [Fonseca et al. 2015]. Já o segundo cenário, usa os conceitos de TL para reutilizar comprimentos de onda em uso na rede e que tenham bandas disponíveis para alocação da requisição. Uma vez que é montada a TL da rede MORFEUS, como mostrado na Figura 2, a heurística FF é aplicada dentre o conjunto de comprimentos de onda e bandas livres na TL para tentar reaproveitar um comprimento de onda já em uso na rede. Caso não seja possível alocar a requisição usando a TL, como pode ser visto na Figura 2, é feita uma busca por uma rota, comprimento de onda e banda ainda livres na rede para atendimento da requisição.

Além da FF, a tradicional heurística Random (RD) e uma nova heurística Extremes (Ext), que foi apresentada em [Fonseca et al. 2015], foram investigadas. De maneira similar aos resultados apresentados em [Fonseca et al. 2015], o desempenho da heurística RD foi bem inferior ao da FF, mesmo no cenário com TL, e, portanto, não são mostrados neste trabalho. Já a heurística Ext apresentou resultados similares a FF no cenário com o uso da TL e, de forma análoga, os resultados não são apresentados neste artigo por uma questão de melhor visualização das curvas no gráficos apresentados na seção de resultados.

Por fim, na sequência do texto, as heurísticas usadas são identificadas por XX-YY, o qual XX e YY informa, respectivamente, as heurísticas WA e BA. O uso da topologia lógica é representado por TL. Portanto, como apontado anteriormente, neste trabalho foram investigados FF-FF e TL-FF-FF.

4. Resultados

4.1. Cenário de Simulação

Baseado em [Fonseca et. al. 2015], foi feita uma análise de uma rede MORFEUS em anel bidirecional com 9 nós e com capacidade total de $N_{B,T} = 72$ e 360 bandas, com cada comprimento de onda contendo 3, 6 e 9 bandas. Por exemplo, a rede com uma capacidade total de 72 bandas podem ter as seguintes possibilidades: i) $W = 24$ comprimentos de onda com $N_B = 3$ bandas cada; ii) $W = 12$ comprimentos de onda com $N_B = 6$ bandas cada; iii) $W = 8$ comprimentos de onda com $N_B = 9$ bandas cada. O caso de 360 bandas é analisado nas simulações numéricas seguindo a mesma escala.

A chegada de requisições na rede são assumidas como Poissonianas, com tráfego uniforme entre os pares de nós do anel bidirecional e cada requisição pode gerar uma demanda com 1, 2 ou 3 bandas. A duração das conexões é exponencialmente distribuída. Considera-se também que cada banda do sinal MB-OFDM possui capacidade de 10 Gbit/s, o que corresponde a dizer que a demanda dos clientes são de 10, 20 e 30 Gbit/s e redes com capacidade total de 720 Gbit/s, 3,6 Tbit/s são considerados. Em cada rodada de uma simulação são geradas 10^6 requisições.

4.2. Impacto na probabilidade de bloqueio

O desempenho da rede é dado pela probabilidade de bloqueio, a qual significa o número de requisições bloqueadas pelo algoritmo RWBA dentre o número total de requisições simuladas. Na Figura 4 os gráficos (a) e (b) mostram a probabilidade de bloqueio para um anel de 9 nós e $N_{B,T} = 72$ e 360 bandas, respectivamente, comparando as heurísticas VT-FF-FF e FF-FF. Como pode-se perceber, a performance da heurística FF-FF é superior a

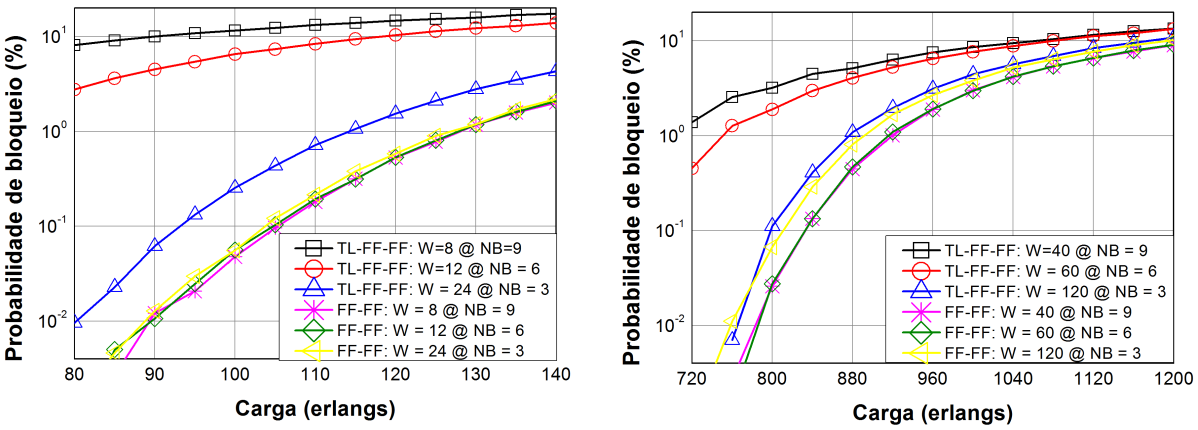


Figura 4. Probabilidade de bloqueio para a rede de 9 nós comparando FF-FF e TL-FF-FF com (a) 72 bandas e (b) 360 bandas.

VT-FF-FF na maioria dos casos, porém para os casos em que cada comprimento de onda possui N_B =3 bandas, a diferença de desempenho é pequena ou não existe diferença. Como visto anteriormente, o uso da topologia lógica para reaproveitar as conexões já estabelecidas na rede pode gerar um bloqueio de futuras requisições, já que para uma requisição entre o nós **i** e **j** ser estabelecida em uma dada conexão **C** já existente na rede, é necessário que a conexão **C** tenha os mesmos nós **i** e **j** como origem e destino; ou duas ou mais conexões são aproveitadas, sendo que o nó de origem da primeira é o nó **i**, e o nó de destino da última conexão é o nó **j**. Com isso o algoritmo VT-FF-FF acaba gerando mais bloqueios que o FF-FF, pois o FF-FF não possui essa restrição, como está detalhado em [Fonseca et al. 2015]. Essa diferença de desempenho diminui sensivelmente, a medida que a capacidade da rede passa a ser distribuída em mais comprimentos de onda, ou seja, a rede passa a utilizar menos bandas por comprimento de onda e, portanto, mais comprimentos de onda. Por exemplo, para a rede MORFEUS com 9 nós e operando com $N_{B,T} = 72$ bandas, a diferença na probabilidade de bloqueio entre FF-FF e VT-FF-FF está próxima de uma ordem de magnitude para um baixo tráfego (90 E) e mantém a média na região de maior tráfego (120 E). Resultados similares são encontrados para a rede com 360 bandas, como visto na Figura 4(b). A exceção é para o caso com N_B =3 bandas por comprimento de onda.

Um ponto importante a ser percebido é que uma vez que o FF-FF prioriza a utilização de bandas do mesmo comprimento de onda, é evidente que, com o aumento do número de bandas por comprimento de onda, existe uma maior liberdade de atribuir faixas contíguas a novos pedidos de conexão. Já o algoritmo TL-FF-FF se beneficia pouco deste fato, pois só pode reaproveitar conexões que ligam os mesmos pares de nós de uma requisição em análise. Assim, se a rede opera com N_B =9 bandas por comprimento de onda, são necessários 40 comprimentos de onda para uma capacidade total da rede de 360 bandas; com N_B =3 bandas são necessários 120 comprimentos de onda, o que passa a ser vantajoso para o algoritmo TL-FF-FF do ponto de vista do desempenho de bloqueio de conexões, como foi confirmado pelas simulações numéricas apresentadas na Figura 4. Na próxima subseção será mostrado que isso também gera uma outra vantagem para o algoritmo TL-FF-FF e para o custo total da rede WDM MB-OFDM.

4.3. Impacto no uso de comprimentos de onda e blocos MIB/MEB‘s

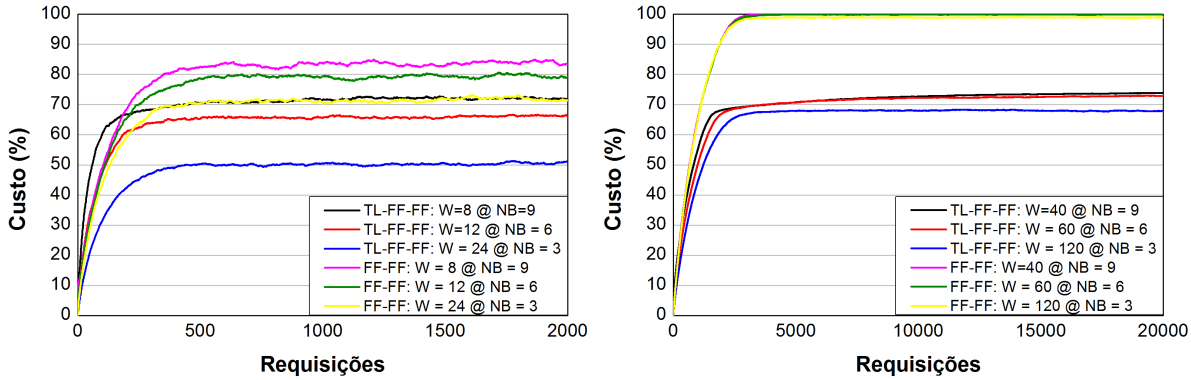


Figura 5. Custo para a rede de 9 nós comparando FF-FF e TL-FF-FF com (a) 72 bandas e (b) 360 bandas.

A Figura 5 mostra o custo da rede MORFEUS com a capacidade total de 72 e 360 bandas para um tráfego de 100 E e 1000 E, respectivamente. Os gráficos foram obtidos em um conjunto de 100 simulações independentes. O eixo vertical foi calculado usando a função da equação (1), a qual representa o uso de comprimentos de onda e blocos MEB/MIB‘s na rede

$$\frac{\sum_{i \in N} \sum_{j \in W} c(i,j)}{W_{total} \times N_{total}}, \tag{1}$$

em que N_{total} é o número de nós, W_{total} o número total de comprimento da onda e $c(i,j)$ representa o uso de MIB's e MEB's indexada pelo número do enlace i e pelo comprimento de onda j. O valor $c(i,j) = 1$ indica que o comprimento de onda j do enlace i está ocupado, enquanto que o valor $c(i,j) = 0$, qualquer que seja o enlace i, indica que o comprimento de onda j está livre.

Como pode ser visto na Figura 5(a), a partir da requisição 500, as curvas tendem a se estabilizar e mostram o valor custo com o uso de MIB's e MEB's. Como previsto, como com a utilização da topologia lógica, primeiro procura-se alocar as bandas em comprimentos de onda com caminhos ópticos já existentes, o custo da rede tende a ser menor quando o algorimo RWBA TL-FF-FF é utilizado. Por exemplo, o cenário de rede com capacidade 72 bandas, a heurística VT-FF-FF e com N_B =3 bandas por comprimento de onda teve uma maior economia no uso de MIB's e MEB's, apresentando uma média de 50% de uso; enquanto o FF-FF apresentou 70% de uso neste mesmo cenário, o que corresponde a um aumento de 40% no uso de comprimentos de onda e MIB/MEB‘s. Resultados similares são obtidos para o cenário de rede com capacidade de 360 bandas. As curvas com a heurística FF-FF continuam com uma grande diferença no custo em comparação com o TL-FF-FF. Com o aumento de tráfego e o número de bandas as curvas tendem a estacionar a um custo constante. Para o RWBA TL-FF-FF, o custo médio a partir da requisição 4000 estaciona em 72%, enquanto que a heurística FF-FF agora tem um custo maximo da rede em 100%.

Um fator importante a ser observado entre os gráficos na Figura 5(a) e (b) é mostrar a relação do aumento no número de comprimentos de onda com a aproximação das

curvas de custo com as heurísticas FF-FF; e o mesmo acontece com TL-FF-FF. O custo no cenário de rede com capacidade total de 72 bandas depende diretamente da forma como é configurada em relação ao número de bandas por comprimento de onda, apenas as curvas do FF-FF com 3 bandas por comprimento de onda e TL-FF-FF com 9 bandas por comprimento de onda estacionam em valores próximos, enquanto que as outras configurações da rede possuem comportamentos distintos no custo de MIB/MEB's. Observe também que o cenário de rede configurado com N_B =3 bandas por comprimento de onda e executando o RWBA TL-FF-FF obteve a melhor economia no uso de MIB/MEB's e comprimentos de onda para ambas as capacidades 72 e 360 bandas. Finalmente, lembre-se que esse cenário de rede também foi o que apresentou probabilidade de bloqueio similar entre o FF-FF e o TF-FF-FF, portanto, os resultados das simulações numéricas sugerem que é mais vantajoso utilizar o TL-FF-FF já que apresenta bloqueio similar ao FF-FF, mas considerável diferença no custo de MIB/MEB's e comprimentos de onda.

4.4. Impacto do planejamento de capacidade WDM MB-OFDM no desempenho da rede

Linha	*Capacidade da rede ($N_{B,T}$)*	*RWBA*	*PB (%)*	*Custo (E)*
1	72 bandas	FF-FF: W = 8 @ NB = 9	0,0478	83%
2		FF-FF: W = 12 @ NB = 6	0,056	79%
3		FF-FF: W = 24 @ NB = 3	0,0558	71%
4		TL-FF-FF: W = 8 @ NB = 9	11,5987	72%
5		TL-FF-FF: W = 12 @ NB = 6	6,517	66%
6		TL-FF-FF: W = 24 @ NB = 3	0,2542	50%
7	360 bandas	FF-FF: W = 40 @ NB = 9	3,0114	100%
8		FF-FF: W = 60 @ NB = 6	2,9747	100%
9		FF-FF: W = 120 @ NB = 3	3,8238	99%
10		TL-FF-FF: W = 40 @ NB = 9	8,5907	72%
11		TL-FF-FF: W = 60 @ NB = 6	7,664	72%
12		TL-FF-FF: W = 120 @ NB = 3	4,1542	68%

Tabela 1. Rede anel com N = 9 e tráfego de 100E para 72 bandas e 1000E para 360 bandas.

Uma discussão interessante ocorre quando se analisa o impacto do planejamento de capacidade WDM MB-OFDM no desempenho da rede. Algoritmos RWBA apresentam diferentes probabilidades de bloqueio dependendo do número de comprimentos de onda e o número de bandas por comprimento de onda adotados. Suponha que o operador da rede queira implantar uma rede com um pico de usuários um tráfego máximo de 100 E para uma rede com baixa capacidade. Quando a Tabela 1 é analisada pode-se perceber que FF-FF com W = 8 e NB = 9 tem o melhor desempenho mas um maior gasto com 83% dos comprimentos de onda em uso, o TL-FF-FF com W = 24 e NB = 3 tem o menor custo com 50% e um desempenho na probabilidade de bloqueio próximo ao FF-FF com 0,2542. Para uma rede de alta capacidade com um tráfego máximo de 1000 E. Pode-se perceber que o desempenho da rede com W = 60 e W = 40 estão muito próximos na probabilidade de bloqueio e no custo elas irão operar com o uso de 100% dos comprimentos de onda, enquanto que o uso do RWBA TL-FF-FF com W = 24 tem uma probabilidade de bloqueio de 4,1% e 47% a menos no uso dos comprimentos de onda e MIB's/MEB's da rede. Com isso, é possível avaliar qual o tipo de arquitetura implantar na rede dependendo de suas necessidades.

5. Conclusões

O problema RWBA recentemente proposto para sinais MB-OFDM foi investigado utilizando topologias lógicas para minimizar o uso de comprimentos de onda e MIB's/MEB's de cada nó. É possível verificar que o projeto apropriado da configuração do número de bandas por comprimento de onda pode levar a resultados promissores do ponto de vista de custo e de desempenho de bloqueio, concomitantemente. A inclusão de efeitos da camada física, bem como o desenvolvimento de novas heurísticas para o problema RWBA estão atualmente em análise.

6. Agradecimentos

Este trabalho é apoiado pela Fundação para a Ciência e Tecnologia de Portugal, e pela CAPES e CNPq do Brasil.

Referências

Alves, T., Alberto, A., and Cartaxo, A. V. T. (2014). Direct-detection multi-band ofdm metro networks employing virtual carriers and low receiver bandwidth. In *Proc. of Optical Fiber Communication Conference and Exposition (OFC 2014)*, volume Tu, page 5.

Alves, T. M. F., Mendes, L. M. M., and Cartaxo, A. V. T. (2015). High granularity multiband ofdm virtual carrier-assisted direct-detection metro networks. *IEEE/OSA Journal of Lightwave Technology*, 33(1):42–54.

Blouza, S., Karaki, J., Brochier, N., Rouzic, E. L., and E. Pincemin, B. C. (2011). Multi-band ofdm for optical networking. *IEEE EUROCON 2011 - International Conference on Computer as a Tool*, pages 1–4.

Cartaxo, A. V. T., Alves, T. M. F., and Mendes, L. M. M. (2014). 42.8 gb/s ssb dd mb-ofdm metro networks assisted by virtual carriers: system parameters optimization. We(A1):1.

Fonseca, I. E., Gama, E. S., Oliveira, C., Júnior, R. A., Alves, T., Rosário, J., and Cartaxo, A. (2015). The new problem of routing, wavelength and band assignment in mb-ofdm metropolitan networks. In *2015 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC) (IMOC 2015)*.

Li, A., Che, D., Chen, X., Hu, Q., Wang, Y., and Shieh, W. (2013). 61 gbits/s direct-detection optical ofdm based on blockwise signal phase switching with signal-to-signal beat noise cancellation. *Optics Letters*, 38(14):2614–2616.

Mendes, L., Cruz, P. E. D., Alves, T. M. F., and Cartaxo, A. V. T. (2014). Impact of the band selector detuning on dd multi-band ofdm virtual-carrier assisted metro network performance. Tu(A3):4.

Peng, W., Morita, I., Takahashi, H., , and Tsuritani, T. (2012). Transmission of high speed (¿ 100gb/s) direct-detection optical ofdm superchannel. *IEEE/OSA Journal of Lightwave Technology*, 30(12):2025–2034.

Peng, W., Zhang, B., Feng, K., Wu, X., Willner, A. E., and Chi, S. (2009). Spectrally efficient direct-detection ofdm transmission incorporating a tunable frequency gap and an iterative detection techniques. *IEEE/OSA Journal of Lightwave Technology*, 27(24):5723–5735.

Um Novo Algoritmo RSA Ciente de Imperfeições de Camada Física para Redes Ópticas Elásticas

Alexandre Fontinele[1], Iallen Santos[2], Juarez Nolêto Neto[1], Divanilson R. Campelo[3] e André Soares[1]

[1] Departamento de Computação
Universidade Federal do Piauí (UFPI)
Teresina – PI – Brasil

[2]Instituto Federal de Educação, Ciência e Tecnologia do Piauí (IFPI)
São Raimundo Nonato – PI – Brasil

[3]Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brasil

andre.soares@ufpi.edu.br

Abstract. *This paper proposes a new routing and spectrum assignment algorithm that considers the impact of physical layer effects for elastic optical networks. The goal of the algorithm is to reduce the blocking probability caused by the degradation of the transmission quality from new establishments of circuits. Our proposal is compared to two other Impairment-Aware Routing and Spectrum Assignment (IA-RSA) algorithms: Modified Dijkstra Path Computation (MD-PC) e K-Shortest Path Computation (KS-PC). Simulation results show that the proposed algorithm outperforms the MD-PC and KS-PC algorithms in terms of circuit blocking probability and bandwidth blocking probability for the EON and NSFNet topologies. In general, the proposed algorithm achieves a minimum gain of 78.10% and 55.75% when compared with the KS-PC and MD-PC algorithms, respectively. In terms of bandwidth blocking probability, the proposed algorithm presents a minimum gain of 71.96% and 41.89% in relation to the KS-PC and MD-PC algorithms, respectively.*

Resumo. *Este artigo propõe um novo algoritmo de roteamento e alocação de espectro ciente dos efeitos de camada física para redes ópticas elásticas. O objetivo do algoritmo é reduzir a probabilidade de bloqueio causada pela degradação da qualidade de transmissão proveniente de novos estabelecimentos de circuitos. A nossa proposta é comparada com outros dois algoritmos Impairment-Aware Routing and Spectrum Assignment (IA-RSA): Modified Dijkstra Path Computation (MD-PC) e K-Shortest Path Computation (KS-PC). Resultados de simulação mostram que o algoritmo proposto apresenta um desempenho superior ao dos algoritmos MD-PC e KS-PC em termos de probabilidade de bloqueio de circuitos e probabilidade de bloqueio de banda para as topologias EON e NSFNet. Em geral, em termos de probabilidade de bloqueio de circuitos o algoritmo proposto apresenta um ganho mínimo de 78,10% e 55,75% quando comparado aos algoritmos KS-PC e MD-PC, respectivamente.*

Em termos de probabilidade de bloqueio de banda, o algoritmo proposto apresenta um ganho mínimo de 71,96% e 41,89% em relação aos algoritmos KS-PC e MD-PC, respectivamente.

1. Introdução

As soluções de transporte de dados baseadas em rede óptica têm se mostrado promissoras para o futuro das comunicações ópticas de alta velocidade [Chatterjee et al. 2015]. Nas últimas décadas, as soluções de redes ópticas de transporte implantadas e estudadas na literatura eram fortemente baseadas na tecnologia WDM (*Wavelength Division Multiplexing*) [Nath et al. 2014].

Em uma rede óptica WDM, o espectro é divido em comprimentos de onda de tamanho fixo [Rahbar 2012, Chatterjee et al. 2015]. A utilização de comprimentos de onda de tamanho fixo provoca a subutilização dos recursos da rede, pois cada requisição de circuito pode requerer diferentes larguras de banda.

Devido à essa deficiência, o uso da tecnologia OFDM (*Ortogonal Frequency Division Multiplexing*) foi proposto recentemente. Uma rede óptica OFDM tem o potencial de alocar espectro para os circuitos ópticos de acordo com os requisitos de largura de banda. Com a tecnologia OFDM o espectro óptico é dividido em slots de frequência (12,5 GHz) e para cada circuito pode ser alocado uma quantidade diferente de slots [Chatterjee et al. 2015]. A utilização do espectro óptico em uma rede óptica OFDM é mais eficiente quando comparada a de uma rede óptica WDM. Essa flexibilidade de alocação de largura de banda das redes ópticas OFDM deu origem a uma nova geração de redes ópticas de transporte, chamada de redes ópticas elásticas.

Nas redes ópticas elásticas transparentes, objeto de investigação deste trabalho, os dados são transmitidos da fonte para o destino como um sinal óptico. Toda a operação de roteamento é realizada no domínio óptico sem qualquer conversão para o domínio eletrônico. Para o estabelecimento de um circuito óptico em uma rede óptica elástica é necessário resolver o problema de roteamento e alocação de espectro (RSA – *Routing and Spectrum Assignment*) [Chatterjee et al. 2015]. O problema RSA consiste em definir uma rota para um par de nós origem e destino. Em seguida, a solução RSA deve escolher uma faixa de espectro livre (conjunto de slots de frequência contíguos) na rota definida para estabelecer o circuito óptico.

O sinal óptico sofre degradação à medida que se propaga da origem para o destino devido às imperfeições da camada física. O acúmulo das degradações no sinal óptico ao longo de uma rota pode tornar a taxa de erro de bit (BER – *Bit Error Rate*) intolerável no receptor, inviabilizando a comunicação [Rahbar 2012]. Nesse contexto, algoritmos RSA que levam em consideração as imperfeições de camada física (IA-RSA – *Impairment-Aware RSA*) tornam-se mais apropriados para encontrar uma solução para o problema de roteamento e alocação de espectro.

Este artigo apresenta um novo algoritmo de roteamento e alocação de espectro ciente de imperfeições da camada física em redes ópticas elásticas. O algoritmo proposto busca estabelecer novos circuitos reduzindo o impacto na qualidade de transmissão (QoT – *Quality of Transmission*) dos outros circuitos já ativos na rede. É também apresentado um estudo de avaliação de desempenho comparando o algoritmo proposto com outros

dois algoritmos IS-RSA da literatura. Os resultados mostram um desempenho superior do algoritmo proposto em relação aos outros algoritmos em termos de probabilidade de bloqueio de circuitos e probabilidade de bloqueio de banda.

As demais seções deste artigo estão organizadas da seguinte forma. A Seção 2 apresenta as redes ópticas elásticas e o problema de roteamento e alocação de espectro. A Seção 3 discute os trabalhos relacionados e apresenta as contribuições deste artigo. O modelo de camada física utilizado para computar a qualidade da transmissão é descrito na Seção 4. O algoritmo proposto é apresentado na Seção 5. A Seção 6 apresenta um estudo de avaliação de desempenho considerando as métricas de probabilidade de bloqueio de circuitos e probabilidade de bloqueio de banda. Por fim, as conclusões do trabalho são destacadas na Seção 7.

2. Redes Ópticas Elásticas

A tecnologia OFDM divide o espectro em pequenos intervalos de frequência, chamado de slots de frequência. Um *transponder* OFDM gera um sinal óptico utilizando apenas a quantidade de slots suficientes, com o apropriado nível de modulação, para satisfazer os requerimentos da requisição. Cada circuito é separado dos demais por um intervalo de frequência, denominado de banda de guarda, de forma a evitar interferências entras as transmissões simultâneas. Essas redes são chamadas de redes ópticas elásticas OFDM [Chatterjee et al. 2015].

A Figura 1 mostra a arquitetura típica da rede óptica elástica, que consiste principalmente de BVT (*Bandwidth Variable Transponder*) e BV-WXC (*Bandwidth Variable Cross-connect*). BVTs são utilizados para adaptar a largura de banda através do ajuste da taxa de bits da transmissão ou formato de modulação. BVTs dão suporte a tranmissões de alta velocidade usando formatos de modulação mais eficientes, como, por exemplo, a utilização do formato de modulação 64-QAM (*Quadrature Amplitude Modulation*) para circuitos ópticos de curta distância. Para circuitos ópticos de longa distância são utilizados formatos de modulação mais robustos, porém menos eficientes em termos de alocação de espectro, tais como os formatos de modulação QPSK (*Quadrature Phase-Shift Keying*) ou BPSK (*Binary Phase-Shift Keying*). Portanto, BVTs são capazes de negociar eficiência espectral para um alcance da transmissão.

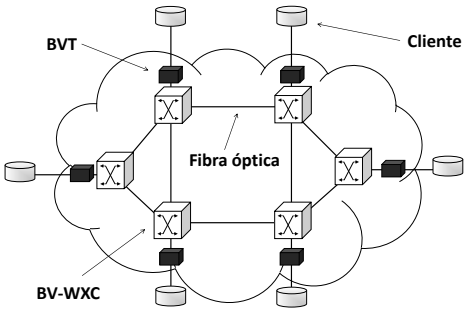


Figura 1. Arquitetura da rede óptica elástica (Adaptada de [Chatterjee et al. 2015]).

Em uma rede óptica, para que os seus clientes se comuniquem uns com os outros, é necessário que sejam estabelecidos circuitos ópticos. Em geral, as requisições de circuitos

podem ser caracterizadas por dois tipos de tráfego [Rahbar 2012]: estático ou dinâmico. No tráfego estático, as requisições são conhecidas previamente e o problema consiste em organizá-las de tal maneira que se minimize a quantidade total dos recursos (slosts de frequência e/ou fibras) a serem implantados na rede. No tráfego dinâmico, considerado neste artigo, as requisições chegam aleatoriamente e são desativadas após um período de tempo.

Depois do processo de admissão de uma requisição, o circuito óptico é estabelecido a partir de um BVT no nó de origem. Este circuito atravessa de forma transparente os BV-WXC nos nós intermediários, e termina em um BVT no nó de destino. Um BV-SSS (*Bandwidth Variable Spectrum Selective Switches*) pode ser utilizado para implementar um BV-WXC [Chatterjee et al. 2015]. Normalmente, um BV-SSS realiza multiplexação/demultiplexação de espectro e funções de comutação óptica. O BV-WXC configura a sua janela de comutação de forma flexível de acordo com a largura espectral do sinal óptico de entrada.

O algoritmo RSA é usado para encontrar uma rota apropriada para um par origem e destino e alocar um conjunto adequado de slots de frequência para o circuito óptico requisitado [Chatterjee et al. 2015]. Estes slots de frequência devem ser alocados adjacentes uns aos outros para atender a restrição de contiguidade de espectro. Além disso, a continuidade desses slots de frequência em cada enlace da rota deverá ser garantida, de forma semelhante à restrição de continuidade de comprimento de onda. Se a requisição solicita t unidades de espectro, então t slots de frequência contíguos devem ser alocados a ela (devido à restrição de contiguidade de espectro). Os mesmos t slots contíguos devem ser alocados em cada enlace ao longo da rota da demanda (devido à restrição de continuidade do espectro).

A dinâmica de alocação de recursos espectrais para as requisições de circuitos e sua liberação após o término das transmissões pode provocar a fragmentação do espectro, problema que influencia diretamente na eficiência das redes ópticas elásticas OFDM. A fragmentação, neste caso, refere-se à existência de pequenos blocos isolados, formados por um ou mais slots contíguos de espectro, que podem não ser suficientes para acomodar futuras requisições de circuitos. Para superar o problema de fragmentação do espectro, diferentes abordagens [Chatterjee et al. 2015] têm sido propostas.

3. Trabalhos Relacionados

A maioria dos trabalhos existentes em redes ópticas elásticas utiliza um limiar de alcance de transmissão [Jinno et al. 2010, Christodoulopoulos et al. 2011, Wan et al. 2012, Zhou et al. 2012, Zhu et al. 2013, Gong et al. 2013, Izquierdo-Zaragoza et al. 2014, Chatterjee et al. 2015] relacionado a cada formato de modulação. Também é utilizada uma banda de guarda para evitar a interferência entre circuitos adjacentes. Entretanto, esses métodos podem superestimar ou subestimar os impactos das imperfeições de camada física nas conexões [Zhao et al. 2015].

O conceito de algoritmos RSA cientes de imperfeições de camada física (IA-RSA – *Impairment Aware Routing and Spectrum Assignment algorithms*) é importante devido aos efeitos prejudiciais da coexistência de vários circuitos ópticos com diferentes formatos de modulações e taxas de bits [Beyranvand and Salehi 2013]. Também é importante escolher o formato de modulação adequado para um circuito através de algoritmos

RSA cientes de qualidade transmissão (QoT-Aware RSA) [Beyranvand and Salehi 2013, Chatterjee et al. 2015].

Em [Yang and Kuipers 2012] é apresentado um algoritmo de RSA ciente da QoT para redes ópticas elásticas translúcidas. O trabalho não aborda os efeitos de camada física que prejudicam a QoT, apenas considera que o alcance da transmissão do sinal está relacionado com o formato de modulação utilizado. Assim, o sinal dever ser regenerado quando o circuito percorrer uma distância maior que o alcance de transmissão permitido pelo formato de modulação selecionado. É utilizado o algoritmo de menor caminho de Dijkstra com restrições extras para computar os caminhos viáveis. As restrições buscam caminhos com a disponibilidade de slots livres em comum, distância máxima de transmissão e uso de regeneradores disponíveis.

Os autores em [Zhao et al. 2015] estudam o problema RSA para o estabelecimento de circuitos off-line em redes ópticas elásticas OFDM. Ainda em [Zhao et al. 2015], os autores propõem uma nova formulação de programação linear inteira e duas heurísticas de RSA cientes de efeitos de camada física, chamadas GL (*Group List*) e CL (*Connection List*). As duas heurísticas ordenam de forma decrescente os circuitos pelos seus requerimentos de taxa de bits. Na GL, os circuitos são agrupados, e depois é aplicada a programação linear inteira sobre cada grupo de circuitos para estabelecer as requisições. A CL considera os circuitos um por vez e combina o uso do algoritmo de menor caminho de Dijkstra para estabelecer os circuitos. Ambas as heurísticas utilizam uma margem de relação sinal ruído (SNR, *Signal to Noise Ratio*) para proteger os circuitos de interferências de uns nos outros. No momento que as heurísticas são aplicadas, essa margem vai sendo adaptada para permitir o estabelecimento de mais circuitos. É utilizado um modelo analítico, proposto em [Johannisson and Agrell 2014], para calcular os efeitos de camada física para cada circuitos.

Os autores em [Beyranvand and Salehi 2013] propõem um esquema de RSA ciente da qualidade de transmissão (QoT) para redes ópticas elásticas. O esquema emprega três etapas: i) cálculo dos rotas viáveis, ii) seleção de uma rota e iii) alocação de espectro. O artigo apresenta um modelo analítico para calcular o SNR de uma dada rota. Baseado no esquema proposto são apresentados dois algortimos IA-RSA: *Modified Dijkstra Path Computation* (MD-PC) e *K-Shortest Path Computation* (KS-PC).

O algoritmo MD-PC é uma versão modificada do algoritmo de menor caminho de Dijkstra. O MD-PC computa todos as rotas possíveis entre um dado par origem e destino, e determina o formato de modulação adequado para cada rota. No momento que um nó é visitado, o algoritmo só marca para serem visitados os vizinhos deste nó em que os enlaces possuem espectro livre contínuo que dão suporte à taxa de bits requisitada.

No algoritmo KS-PC, o algoritmo de k-menores caminhos de Yen é utilizado para obter as k rotas candidatas. Para cada rota do conjunto de rotas candidatas é determinado um formato de modulação adequado. Se não for possível encontrar um formato de modulação para uma dada rota, esta rota é retirada do conjunto de rotas candidatas. A próxima etapa é determinar a rota mais qualificada, referenciada como rota dominante, em termos das métricas de QoT e eficiência espectral.

Algoritmos RSA cientes de QoT devem garantir a qualidade do sinal de um circuito em estabelecimento sem interromper os circuitos já presentes na rede. Essa

preocupação com a QoT dos circuitos já ativos na rede ainda é pouco estudada em redes ópticas elásticas [Chatterjee et al. 2015, Zhao et al. 2015]. A principal contribuição deste artigo é a proposta de um novo algoritmo de roteamento e alocação de espectro ciente de efeitos de camada física em redes ópticas elásticas sob tráfego dinâmico. O principal diferencial do algoritmo proposto em relação às outras propostas presentes na literatura são as estratégias utilizadas para as escolhas do formato de modulação e da rota para cada circuito óptico em estabelecimento. Essas estratégias buscam estabelecer circuitos mais resistentes às degradações dos efeitos de camada física e que reduzam o impacto na QoT dos outros circuitos já ativos na rede.

4. Modelo da Camada Física

Em uma rede óptica elástica OFDM, os dados são transmitidos por n slots de frequência modulados para se criar um circuito óptico. O alcance de transmissão do nível de modulação utilizado é determinado pela SNR correspondente a uma taxa de erro de bit (BER – *Bit Error Rate*) aceitável, por exemplo 10^{-3} [Beyranvand and Salehi 2013].

A Figura 2 apresenta a arquitetura assumida para uma rota com um salto (*span*). Da esquerda para a direita temos: laser transmissor no BVT do nó de origem (A), BV-WXC (onde há um *splitter* e um BV-SSS) do nó de origem (B), fibra óptica (C), amplificador óptico (D), n spans (E), BV-WXC (onde há um *splitter* e um BV-SSS) do nó de destino (F) e receptor no BVT do nó de destino (G).

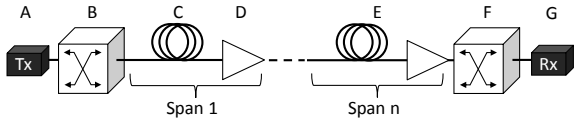


Figura 2. Arquitetura assumida para uma rota com um salto.

Na região linear do OFDM óptico, o alcance de transmissão é limitado pelo ruído ASE (*Amplified Spontaneous Emission*), enquanto na região não linear ele é limitado pelas não linearidades da fibra tais como *Four-Wave Mixing* (mistura de quatro ondas), *Cross-Phase Modulation* (modulação cruzada de fase) e *Self-Phase-Modulation* (auto-modulação de fase) [Beyranvand and Salehi 2013].

Para calcular a SNR de um circuito, foi adotado o modelo de camada física apresentado em [Johannisson and Agrell 2014, Zhao et al. 2015]. O cálculo da SNR para um circuito i usando uma rota r_i é expresso por:

$$SNR_i = \frac{I}{I_{ASE} + I_{NLI}}. \tag{1}$$

A variável I é a densidade espectral da potência do sinal (PSD – *Power Spectral Density*), $I = P_{TX}/\Delta_f$, em que P_{TX} é potência de sinal e Δ_f é a largura de banda do circuito. A PSD do ruído ASE é dada por:

$$I_{ASE} = \sum_{l \in r_i} N_l I^0_{ASE}, \tag{2}$$

em que N_l é o número de spans do enlace l e $I^0_{ASE} = (G_{AMP} - 1)Fhv$. A variável F é o fator de emissão espontânea, que corresponde à metade da figura de ruído (NF – *Noise*

Figure) do amplificador [Beyranvand and Salehi 2013], h é a constante de Planck, v é a frequência da luz e G_{AMP} é o ganho do amplificador óptico. A PSD do ruído dos efeitos não lineares (NLI – *Nonlinear Impairments*) é dada por:

$$I_{NLI} = \sum_{l \in r_i} N_l I^l_{NLI}, \tag{3}$$

em que I^l_{NLI}, a PSD do ruído NLI em um único span do enlace l, é expressa pela Equação 4 [Zhao et al. 2015]:

$$I^l_{NLI} = \frac{3\gamma^2 I^3}{2\pi\alpha|\beta_2|} \left(asinh\left(\frac{\pi^2|\beta_2|}{2\alpha} B_i^2\right) + \sum_j \ln\left[\left(\Delta_{f_{ij}} + \frac{B_j}{2}\right) / \left(\Delta_{f_{ij}} - \frac{B_j}{2}\right)\right]\right), \tag{4}$$

em que, j é outro circuito usando o enlace l, Bi e Bj são, respectivemente, as larguras de bandas para os circuitos i e j, $\Delta_{f_{ij}}$ é o espaçamento da frequência central entre os circuitos i e j, γ é o coeficiente não linear da fibra, β_2 é o parâmetro de dispersão da fibra e α é a atenuação de potência causada pela fibra.

5. Algoritmo Proposto

Um novo circuito óptico é estabelecido se, e somente se, todas as seguintes exigências forem atendidas: i) Há disponibilidade de espectro óptico na rota escolhida; ii) A QoT é aceitável para o circuito que está sendo estabelecido e iii) A QoT para todos os demais circuitos já ativos na rede é mantida aceitável, considerando que o novo circuito óptico seja atendido. Seguindo esta modelagem, pode-se decompor a probabilidade de bloqueio geral da rede em três componentes: i) Bloqueio pela indisponibilidade de espectro livre que dê suporte à largura de banda requisitada pelo novo circuito óptico – esse tipo de bloqueio pode ocorrer pela fragmentação do espectro ou pela Ausência de Espectro Livre (AEL) na rota selecionada; ii) QoT inadequada para o Novo circuito óptico (QoTN) e iii) QoT inadequada para os Outros circuitos ópticas já ativos na rede (QoTO).

O algoritmo de IA-RSA proposto neste artigo é chamado de k menores caminhos com redução de QoTO (KSP-RQoTO – *K-Shortest Path with Reduction of QoTO*). O algoritmo computa os k menores caminhos para cada par origem e destino da rede em uma fase off-line. No momento que a rede está em operação, o algoritmo proposto busca alocar uma das k rotas alternativas de forma a minimizar o bloqueio do tipo QoTO. O fluxograma ilustrado na Figura 3 apresenta os passos seguidos pelo algoritmo proposto na fase de operação da rede.

O primeiro passo do algoritmo KSP-RQoTO é selecionar um formato de modulação para cada rota candidata. Para cada formato de modulação é verificado se é possível alocar espectro e se a QoT está aceitável. Em caso afirmativo é computado o $\Delta\text{SNR}_{MOD\&REQ}$ para o circuito em estabelecimento. O ΔSNR é a diferença entre o SNR atual do circuito em estabelecimento e o limiar de SNR do formato de modulação. O limiar de SNR representa o valor mínimo de SNR tolerado pelo receptor. O ΔSNR representa a margem residual de SNR do circuito óptico. Quanto menor o ΔSNR de um circuito óptico, mais frágil este circuito óptico está as degradações de camada física. O valor do $\Delta\text{SNR}_{MOD\&REQ}$ é atribuído como avaliação para o formato de modulação em analise.

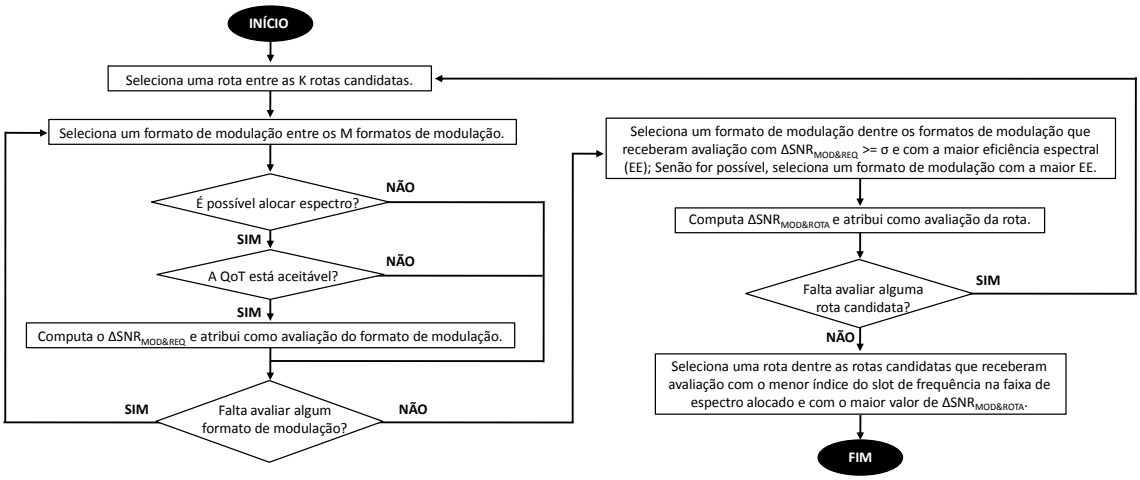


Figura 3. Fluxograma do algoritmo KSP-RQoTO.

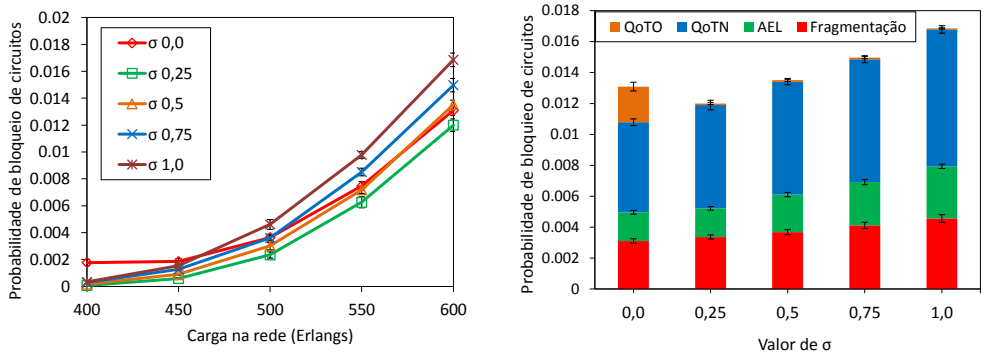
Dentre os formatos de modulação que recebem avaliação, é selecionado o formato de modulação com alto nível que possua o $\Delta\text{SNR}_{MOD\&REQ} \geq \sigma$. σ corresponde a um valor mínimo exigido pelo algoritmo para o ΔSNR de um circuito em estabelecimento adotando um dado formato de modulação. A escolha do valor ideal de σ é apresentada a seguir. Caso nenhum formato de modulação atinja o valor de σ, o algoritmo escolhe aquele formato de modulação que apresentar o maior nível. Formatos de modulação com altos níveis possuem uma maior eficiência espectral que formatos de modulação com baixos níveis. Por outro lado, o alcance de transmissão de formatos de modulação com altos níveis são inferiores ao alcance de transmissão de formatos de modulação com baixos níveis [Chatterjee et al. 2015].

O segundo passo do algoritmo KSP-RQoTO é selecionar uma rota entre as rotas candidatas. Para o processo de escolha da rota é computado o valor de $\Delta\text{SNR}_{MOD\&ROTA}$. Primeiro é calculado o ΔSNR para todos os circuitos ópticos ativos que possuem rotas com enlaces em comum com a rota candidata em avaliação. Para isso o plano de controle da rede mantém para cada circuito óptico ativo a sua rota e o seu SNR calculado no nó destino. O menor ΔSNR entre os ΔSNR calculados para os circuitos ópticos ativos é atribuído como valor para $\Delta\text{SNR}_{MOD\&ROTA}$. O valor de $\Delta\text{SNR}_{MOD\&ROTA}$ é atribuído como avaliação da rota candidata analisada no momento. Rotas candidatas em que nenhum formato de modulação recebeu avaliação são removidas do conjunto de rotas candidatas. A rota escolhida para o estabelecimento do novo circuito óptico será aquela que apresentar o menor índice do slot de frequência na faixa de espectro alocado e maior valor de $\Delta\text{SNR}_{MOD\&ROTA}$ para o circuito em estabelecimento.

Ao escolher a rota em que o índice do slot de frequência na faixa de espectro alocado seja o menor, o algoritmo está tentando acomodar os circuitos mais próximos do início do espectro óptico. Essa estratégia tenta reduzir a fragmentação do espectro [Wang and Mukherjee 2014]. Ao escolher a rota como maior valor de $\Delta\text{SNR}_{MOD\&ROTA}$ neste passo, o algoritmo proposto evita degradar os circuitos ópticos ativos mais frágeis, isto é, os circuitos ativos com o menor ΔSNR. Quanto mais circuitos ópticos frágeis na rede, maior a probabilidade de que as novas requisições sofram bloqueio do tipo QoTO.

A escolha do valor ideal de σ deve ser feita durante a fase de planejamento da rede

ou durante uma alteração da configuração da rede. Para exemplificar a escolha do valor ideal de σ é apresentado um estudo. A Figura 4 apresenta as curvas de probabilidade de bloqueio de circuitos em função da carga na topologia EON (Figura 5) considerando diferentes valores para σ e também apresenta as componentes da probabilidade de bloqueio em função do σ para uma carga de 600 Erlangs. Nesta seção o algoritmo KSP-RQoTO foi aplicado junto do algoritmo de alocação de espectro *First-Fit* [Chatterjee et al. 2015]. Neste cenário (Figura 4(a)), para a carga de 600 Erlangs (carga mais alta estudada) o valor escolhido para o σ é 0,25. Com este valor para o σ a probabilidade de bloqueio apresentou o melhor desempenho. O valor ideal para o σ tem como objetivo estabelecer circuitos mais resistentes as degradações causadas pelo estabelecimento de novos circuitos e deve ser escolhido de maneira a reduzir a probabilidade de bloqueio geral.



(a) Probabilidade de bloqueio geral.

(b) Componentes da probabilidade de bloqueio.

Figura 4. (a) Curvas de probabilidade de bloqueio em função da carga na topologia EON utilizando diferentes valores para σ. (b) Composição da probabilidade de bloqueio geral de acordo com o valor de σ na topologia EON sob uma carga de 600 erlangs.

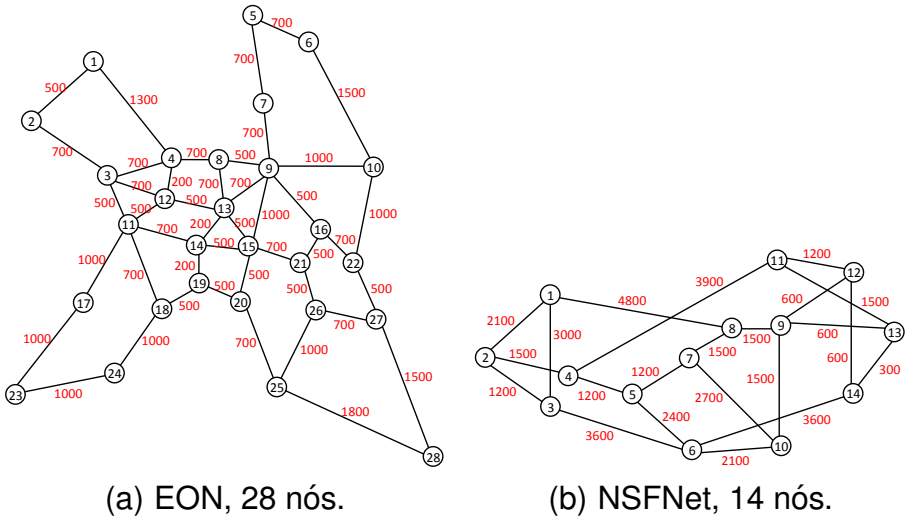
Para definir o valor ideal para o σ, o algoritmo KSP-RQoTO simula o comportamento da rede considerando diferentes valores para σ em ordem crescente. A Figura 4(b) ilustra a busca do valor ideal para o σ sob uma carga de 600 Erlangs. No primeiro momento ($0,0 \leq \sigma < 0,5$), com o aumento do valor de σ a probabilidade de bloqueio vai diminuindo. Os circuitos que estão sendo estabelecidos evitam rotas que causam bloqueio do tipo QoTO. Entretanto, para um dado valor de σ, o comportamento é invertido ($\sigma \geq 0,5$), como é ilustrado na Figura 4(b). Ocorre um aumento na probabilidade de bloqueio. Isso porque estão sendo selecionados formatos de modulação com baixos níveis. Formatos de modulação com baixos níveis tendem a necessitar de mais slots de frequência para estabelecer circuitos. Assim, o algoritmo proposto busca o valor ideal de σ até a inversão no comportamento da curva da probabilidade de bloqueio.

6. Avaliação de Desempenho

O algoritmo proposto, KSP-RQoTO, foi comparado com os algoritmos KS-PC e MD-PC sob os aspectos de probabilidade de bloqueio de circuito e probabilidade de bloqueio de banda. Foram utilizadas quatro rotas condidatas para os algoritmos de k menores caminhos. O algoritmo *First-Fit* foi utilizado para a alocação de espectro

[Chatterjee et al. 2015]. Para avaliar os impactos dos efeitos de camada física nos circuitos ópticos, o modelo de camada física apresentado na Seção 4 foi implementado no SNetS (*SLICE Network Simulator*). O SNetS é uma ferramenta de simulação desenvolvida para permitir a avaliação de desempenho de redes ópticas elásticas OFDM [Santos 2015]. Informações sobre a validação podem ser encontradas em [Santos 2015].

Foram geradas 100000 requisições de circuitos em cada simulação. A geração de requisições é um processo de Poisson com taxa média de λ e o tempo médio de retenção dos circuitos é distribuído exponencialmente com média $1/\mu$. A carga de tráfego é distruída uniformemente entre todos os pares de nós origem e destino. A carga em Erlangs pode ser definida por $\rho = \lambda/\mu$. Para cada simulação foram realizadas 10 replicações com diferentes sementes de geração de variável aleatória. Todos os resultados possuem nível de confiança de 95%. As topologias consideradas nas simulações são EON e NSFNet, Figura 5. O valor apresentado em cada enlace da topologia indica a distância do enlace em km.



(a) EON, 28 nós. (b) NSFNet, 14 nós.

Figura 5. Topologias usadas no estudo de avaliação de desempenho. O número em cada enlace corresponde a distância em km.

Os requisitos de taxas de bits para cada circuito requisitado variam uniformemente entre 10 Gbps, 40 Gbps, 80 Gbps, 100 Gbps, 160 Gbps, 200 Gbps e 400 Gbps. Os formatos de modulação considerados neste estudo foram BPSK, QPSK, 8QAM, 16QAM, 32QAM e 64QAM, seus respectivos limiares de SNR são 6 dB, 9 dB, 12 dB, 15 dB, 18 dB e 21 dB [Beyranvand and Salehi 2013]. Os respectivos níveis dos formatos de modulação são 2, 3, 4, 5, 6 e 7. Para se calcular a largura de banda de uma requisição i para uma dada taxa de bits B_n, com nível de modulação L_m e sobrecarga de FEC (*Forward Error Correction*) F, é utilizada a Equação 5 [Gao et al. 2014]:

$$B_i = \frac{1.1 B_n \left(1 + F\right)}{2 \log_2 L_m}. \tag{5}$$

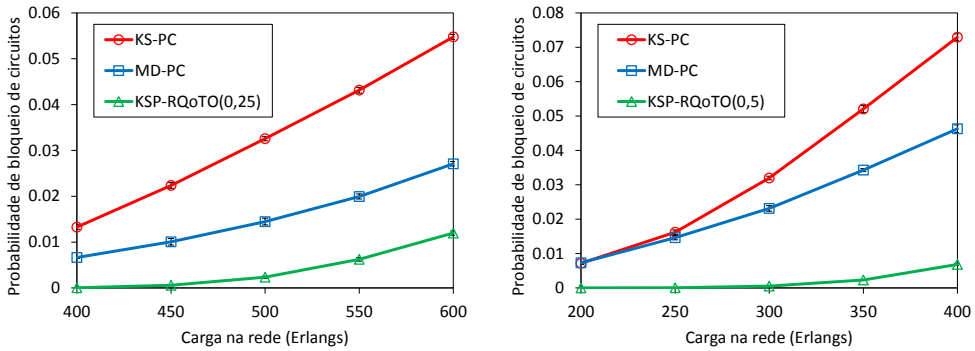
Em seguida, encontra-se um número inteiro de slots de frequência que cubra a largura de banda requisitada acrescentada da banda de guarda. Neste estudo, foi considerada uma FEC de 7%, que corresponde a um limiar de BER de $3,8 \times 10^{-3}$ [Gao et al. 2014].

Todos os enlaces da rede são bidirecionais e possuem largura de banda do espectro dividida em 400 slots de frequência. Um slot de frequência possui largura de banda de 12,5 GHz e a banda de guarda possui largura de banda de 6,25 GHz [Gao et al. 2014]. Os ganhos dos amplificadores são ajustados para compensar as perdas dos dispositivos e da fibra. Outros parâmetros utilizados nas simulações estão listados na Tabela 1 [Beyranvand and Salehi 2013, Zhao et al. 2015].

Tabela 1. Parâmetros de camada física utilizados nas simulações.

Descrição	Valor
Densidade espectral de potência do sinal	-17 dBm/GHz
Atenuaçao da fibra (α)	0,2 dB/km
Parâmetro de dispersão da fibra (β_2)	16 ps^2/km
Coeficiente não linear da fibra (γ)	1,3 (Wkm)$^{-1}$
Tamanho de um span (L_s)	100 km
Figura de ruído do amplificador (NF)	6 dB
Potência de saturação do amplificador (P_{SAT})	16 dBm

A Figura 6 apresenta as probabilidades de bloqueio de circuitos obtidas com os algoritmos KSP-RQoTO, KS-PC e MD-PC em função da carga na rede para as topologias EON e NSFNet. O valor entre parênteses na legenda do algoritmo KSP-RQoTO, ilustrada na Figura 6, corresponde ao valor ideal para o σ. Observando a Figura 6, nota-se que, sob os valores de carga analisados, o algoritmo KSP-RQoTO apresentou melhor desempenho em termos de probabilidade de bloqueio de circuitos do que os algoritmos KS-PC e MD-PC para as duas topologias consideradas.



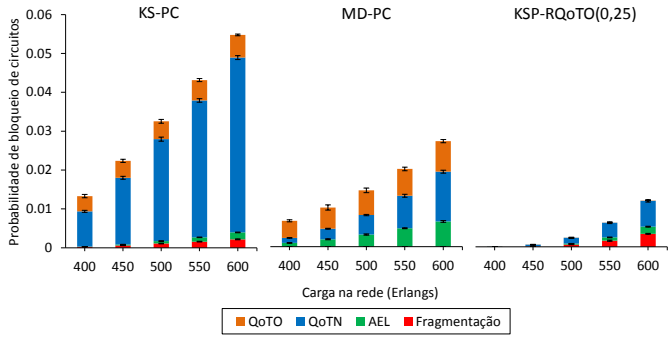
(a) EON. (b) NSFNet.

Figura 6. Probabilidades de bloqueio de circuito obtidas pelos algoritmos KS-PC, MD-PC e KSP-RQoTO paras as topologias (a) EON e (b) NSFNet.

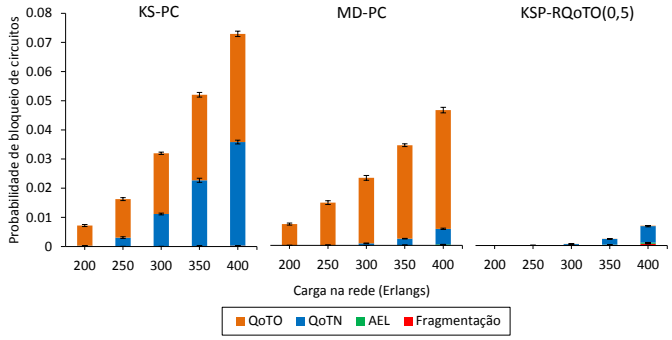
A Figura 6(a) ilustra o desempenho sob uma carga de 600 Erlangs na topologia EON. Pode-se observar que os algoritmos KS-PC, MD-PC e KSP-RQoTO obtiveram 5,48%, 2,71% e 1,20% de probabilidade de bloqueio de circuito, respectivamente. Para esses valores de probabilidades de bloqueio de circuito, o ganho do KSP-RQoTO em relação aos algoritmos KS-PC e MD-PC foi de aproximadamente 78,10% e 55,75%, respectivamente. A Figura Figura 6(b) ilustra o desempenho sob uma carga de 400 Erlangs na topologia NSFNet. Pode-se observar que os algoritmos KS-PC, MD-PC e KSP-RQoTO alcançaram 7,30%, 4,63% e 0,69% de probabilidade de bloqueio de circuito,

respectivamente. O ganho do algoritmo KSP-RQoTO em relação aos algoritmos KS-PC e MD-PC foi de aproximadamente 90,59% e 85,18%, respectivamente.

A Figura 7 exibe as componentes da probabilidade de bloqueio de circuitos obtidas com os algoritmos KSP-RQoTO, KS-PC e MD-PC em função da carga na rede para as topologias EON e NSFNet. O valor entre parênteses na legenda do algoritmo KSP-RQoTO na Figura 7 corresponde ao valor ideal para o σ



(a) EON.



(b) NSFNet.

Figura 7. Componentes das probabilidades de bloqueio de circuito obtidas pelos algoritmos KS-PC, MD-PC e KSP-RQoTO para as topologias (a) EON e (b) NSFNet.

Analisando as componentes da probabilidade de bloqueio quando os algoritmos KS-PC, MD-PC e KSP-RQoTO são aplicados na topologia EON, ilustradas na Figura 7(a), percebe-se que a componente QoTN é a principal responsável pelos bloqueios dos circuitos. Nota-se pela Figura 7(a) que a estratégia do algoritmo KSP-RQoTO em evitar rotas que impactam em circuitos mais frágeis reduz a componente QoTO da probabilidade de bloqueio. Observa-se na Figura 7(b) que as componentes QoTN e QoTO foram as principais responsáveis pelos bloqueios de circuitos quando os algoritmos KS-PC e MD-PC foram aplicados na topologia NSFNet. Nota-se ainda pela Figura 7(b), que o algoritmo KSP-RQoTO reduziu a componente QoTO da probabilidade de bloqueio, assim como ocorreu na topologia EON.

A métrica de probabilidade de bloqueio de circuitos não faz distinção entre o bloqueio de circuitos com requisitos de largura de banda diferentes. Para isso é utilizada a métrica probabilidade de bloqueio de banda. A probabilidade de bloqueio de banda consiste na razão da quantidade de banda bloqueada (somatório da largura de banda vezes o tempo de duração de cada circuito bloqueado) pela quantidade de banda requisitada

(somatório da largura de banda vezes o tempo de duração de cada circuito requisitado).

Devido ao limite de páginas os gráficos de probabilidade de bloqueio de banda não foram inseridos no artigo apenas a análise dos resultados. Para a topologia EON, sob o ponto de carga de 600 Erlangs, os algoritmos KS-PC, MD-PC e KSP-RQoTO alcançaram probabilidade de bloqueio de banda de 10,55%, 5,09% e 2,96%, respectivamente. O ganho do algoritmo KSP-RQoTO em relação aos algoritmos KS-PC e MD-PC foi de aproximadamente 71,96% e 41,89%, respectivamente. Para a topologia NSFNet, sob o ponto de carga de 400 Erlangs, os algoritmos KS-PC, MD-PC e KSP-RQoTO alcançaram probabilidades de bloqueio de banda de 11,72%, 5,76% e 1,73%, respectivamente. O ganho do algoritmo KSP-RQoTO em relação aos algoritmos KS-PC e MD-PC foi de aproximadamente 85,22% e 69,94%, respectivamente. Os resultados obtidos mostram que o algoritmo KSP-RQoTO apresentou o melhor desempenho em termos de probabilidade de bloqueio de banda em relação aos algoritmos KS-PC e MD-PC quanto aplicados às topologias EON e NSFNet.

7. Conclusões

Este artigo propôs o algoritmo KSP-RQoTO que realiza roteamento e alocação de espectro considerando imperfeições de camada física em redes ópticas elásticas. O objetivo do algoritmo KSP-RQotO é estabelecer circuitos que causem o menor impacto na QoT dos outros circuitos já ativos na rede. O algoritmo KSP-RQotO apresentou em desempenho superior ao dos algoritmos KS-PC e MD-PC em termos de probabilidade de bloqueio de circuito e probabilidade de bloqueio de banda nos cenários estudados.

Em termos de probabilidade de bloqueio de circuito, para a topologia EON, o algoritmo KSP-RQotO obteve um ganho de pelo menos 78,10% e 55,75% em relação aos algoritmos KS-PC e MD-PC, respectivamente. Na topologia NSFNet, o algoritmo KSP-RQoTO obteve ganho mínimo de 90,59% e 85,18% em termos de probabilidade de bloqueio de circuitos em realação aos algoritmos KS-PC e MD-PC, respectivamente.

Em termos de probabilidade de bloqueio de banda, para a topologia EON, o algoritmo KSP-RQoTO obteve um ganho de pelo menos 71,96% e 41,89% em relação aos algoritmos KS-PC e MD-PC, respectivamente. Já na topologia NSFNet, em termos de probabilidade de bloqueio de banda, o algoritmo KSP-RQoTO obteve ganho mínimo de 85,22% e 69,94% em relação aos algoritmos KS-PC e MD-PC, respectivamente.

Referências

Beyranvand, H. and Salehi, J. (2013). A quality-of-transmission aware dynamic routing and spectrum assignment scheme for future elastic optical networks. *Journal of Lightwave Technology*, 31(18):3043–3054.

Chatterjee, B. C., Sarma, N., and Oki, E. (2015). Routing and spectrum allocation in elastic optical networks: A tutorial. *IEEE Communications Surveys Tutorials*, 17(3):1776–1800.

Christodoulopoulos, K., Tomkos, I., and Varvarigos, E. (2011). Elastic bandwidth allocation in flexible ofdm-based optical networks. *Journal of Lightwave Technology*, 29(9):1354–1366.

Gao, G., Zhang, J., Wang, L., Gu, W., and Ji, Y. (2014). Influence of physical layer configuration on performance of elastic optical ofdm networks. *IEEE Communications Letters*, 18(4):672–675.

Gong, L., Zhou, X., Liu, X., Zhao, W., Lu, W., and Zhu, Z. (2013). Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 5(8):836–847.

Izquierdo-Zaragoza, J.-L., Pavon-Marino, P., and Bueno-Delgado, M.-V. (2014). Distance-adaptive online rsa algorithms for heterogeneous flex-grid networks. In *2014 International Conference on Optical Network Design and Modeling*, pages 204–209.

Jinno, M., Kozicki, B., Takara, H., Watanabe, A., Sone, Y., Tanaka, T., and Hirano, A. (2010). Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [topics in optical communications]. *IEEE Communications Magazine*, 48(8):138–145.

Johannisson, P. and Agrell, E. (2014). Modeling of nonlinear signal distortion in fiber-optic networks. *Journal of Lightwave Technology*, 32(23):4544–4552.

Nath, I., Chatterjee, M., and Bhattacharya, U. (2014). A survey on regenerator placement problem in translucent optical network. In *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pages 408–413.

Rahbar, A. G. (2012). Review of dynamic impairment-aware routing and wavelength assignment techniques in all-optical wavelength-routed networks. *IEEE Communications Surveys Tutorials*, 14(4):1065–1089.

Santos, I. (2015). *Alocação de Recursos para o Estabelecimento de Circuitos em Redes Ópticas WDM e OFDM*. Universidade Federal do Piauí, Teresina.

Wan, X., Hua, N., and Zheng, X. (2012). Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 4(8):603–613.

Wang, R. and Mukherjee, B. (2014). Spectrum management in heterogeneous bandwidth optical networks. *Optical Switching and Networking*, 11, Part A:83 – 91.

Yang, S. and Kuipers, F. (2012). Impairment-aware routing in translucent spectrum-sliced elastic optical path networks. In *2012 17th European Conference on Networks and Optical Communications (NOC)*, pages 1–6.

Zhao, J., Wymeersch, H., and Agrell, E. (2015). Nonlinear impairment aware resource allocation in elastic optical networks. In *2015 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3.

Zhou, X., Lu, W., Gong, L., and Zhu, Z. (2012). Dynamic rmsa in elastic optical networks with an adaptive genetic algorithm. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 2912–2917.

Zhu, Z., Lu, W., Zhang, L., and Ansari, N. (2013). Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing. *Journal of Lightwave Technology*, 31(1):15–22.

Proteção de Redes Ópticas Elásticas Baseada em Agregação de Tráfego, Sobreposição de Espectro e p-Cycle FIPP

Helder M. N. da S. Oliveira[1], **Nelson L. S. da Fonseca**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas 13083-852, SP, Brasil

helder@lrc.ic.unicamp.br, nfonseca@ic.unicamp.br *

Abstract. *In recent years, the technology of elastic optical networks has emerged as a solution for dealing with the diversity of bandwidth demands of network applications. Therefore, protection techniques have been developed to cope with failures. Among these techniques, p-cycle is a very attractive once since it provides ring-like speed of restoration in mesh topologies. This paper presents a new algorithms to provide path protection using p-cycle path, trafic grooming and overlap espectrum in Flexgrid networks. The proposed algorithm is compared to others in the literature. Results indicate that the our algorithm can provide up to 40% less blocking do existing ones.*

Resumo. *Nos últimos anos, a tecnologia de redes ópticas elásticas tem emergido como uma solução para lidar com a diversidade de requisições de largura de banda das aplicações em rede. No entanto, falhas em enlaces e nós causam perdas maciça de dados, mesmo que por curtos períodos. Portanto, técnicas de proteção têm sido desenvolvidas para lidar com falhas. Entre essas técnicas, p-cycle é muito atraente, uma vez que a capacidade reservada é pré-conectada e que forma estruturas em anel para proteção de redes em malha. Este artigo apresenta um novo algoritmo para fornecer proteção de caminho através do emprego de p-cycle de caminho, agregação de tráfego e sobreposição de espectro em redes ópticas elásticas. O algoritmo proposto é comparado com outros algoritmos existentes na literatura. Os resultados indicam que o algoritmo proposto gera até 40 % menos bloqueio que os outros algoritmos na literatura.*

1. Introdução

O conceito de redes flexíveis, também chamadas redes ópticas elásticas (EONs), foi introduzido em [Jinno et al. 2009]. Nessas redes, o espectro da fibra é tratado como um recurso "contínuo", uma vez que a granularidade de alocação do espectro é fina. Tal alocação difere da alocação grossa adotada nas redes *wavelength division multiplexing* (WDM) possibilitando, assim, uma utilização mais eficiente do espectro. O espectro é dividido em um conjunto de slots tipicamente de 6,25 GHz ou de 12,5 GHz. Caminhos ópticos podem ser alocados em intervalos "contínuo"de frequência e alocações "contínuas"de espectro devem ser separadas por uma banda de guarda, denominada de *Filter Guard Band* (FGB). A introdução dessa nova forma de alocação de espectro demanda novos algoritmos de roteamento, gerenciamento, agrupamento de tráfego, restauração e proteção.

Dada a enorme capacidade das fibras óptica, qualquer interrupção implica em enorme perda de dados. Essa vulnerabilidade tem motivado o desenvolvimento de diferentes esquemas de proteção e restauração. O p-cycle é uma dessas técnicas de

proteção, que tem sido intensivamente investigado nos últimos anos [Asthana et al. 2010] [Schupke 2006] devido às suas vantagens. O *p-cycle* combina as propriedades de velocidade de recuperação em anel com a eficiência das redes em malha restauráveis. Um tipo de *p-cycle* de especial interesse é o *p-cycle* de caminhos independentes de falhas (FIPP) que fornecem caminhos de proteção de redes ópticas totalmente pré-conectados.

A sobreposição de espectro é uma técnica em que dois caminhos ópticos de backup podem utilizar os mesmos enlaces e o mesmo espectro, desde que os caminhos de trabalho das duas conexões sejam enlaces fisicamente disjuntos [Liu et al. 2013]. A sobreposição de espectro permite um ganho significativo na utilização de espectro, o que leva a um menor bloqueio de conexões. A sobreposição do espectro é possível devido a propriedade de elasticidade dos transponders que permite a expansão e contração dos caminhos ópticos. Assim, quando uma falha ocorre na rede, caminhos ópticos que transportam fluxos de backup podem ser ajustados para as taxas apropriadas de modo que os espectros sobrepostos possam ser utilizados por apenas um dos caminhos de backup.

Este artigo introduz um algoritmo chamado FIPPSh-Flex para prover proteção para redes ópticas elásticas. O algoritmo forma caminhos de proteção, utilizando-se da técnica *p-cycle* FIPP, sobreposição de espectro e agregação de tráfego. O algoritmo FIPPSh-Flex estende o algoritmo FIPP-Flex, [Oliveira and da Fonseca 2014] para utilizar a técnica de sobreposição do espectro nos caminhos ópticos de backup (*p-cycles*). A combinação da técnica de sobreposição de espectro, agregação de tráfego e *p-cycle* gera resultados superiores aos demais algoritmos existentes [Shao et al. 2012], [Oliveira and da Fonseca 2014] e [Liu et al. 2013].

Este artigo está organizado da seguinte forma. A seção 2 revisa trabalhos relacionados. A seção 3 apresenta a técnica de agregação de tráfego. A seção 4 introduz os conceitos de *p-cycle* e FIPP. A seção 5 introduz o algoritmo RSA -FLEX e a seção 6 apresenta os algoritmos FIPPSh-FLEX. A seção 7 avalia o desempenho do algoritmo proposto e a seção 8 conclui o artigo.

2. Trabalhos Relacionados

Diversos artigos têm propostos métodos de proteção em redes ópticas elásticas [Liu et al. 2013, Shao et al. 2012, Din and Lai 2015, Oliveira and da Fonseca 2014, Aibin and Walkowiak 2015, Chen et al. 2015]. A seguir serão apresentados breve resumos dos artigos mais relevantes.

Em [Liu et al. 2013], investiga-se o problema de agregação de tráfego de sobrevivência para redes ópticas elásticas com grade flexível do espectro que utilizam a tecnologias de transmissão OFDM. Propõe-se uma abordagem de caminho compartilhado que explora compartilhamento de caminhos de backup, bem como a sobreposição do espectro de *backup* de diferentes caminhos.

Os autores de [Shao et al. 2012] propuseram políticas conservadoras de proteção em redes ópticas elásticas. Eles introduziram uma política em que caminhos ópticos de *backup* protegem caminhos ópticos primários utilizando caminhos disjuntos, levando a maior robustez e proteção.

Em [Din and Lai 2015], estuda-se o problema de proteção *multicast* em redes ópticas elásticas para uma única falha. Dois algoritmos de proteção baseada em

segmentação de caminho foram propostos para resolver este problema.

Em [Oliveira and da Fonseca 2014] introduziram-se dois algoritmos chamados FIPP-Flex e FIPP-Flex-twofailure, para proteção de caminhos através do uso de p-cycle FIPP em redes ópticas elásticas (Flexgrid). O algoritmo FIPP-Flex provê proteção contra uma única falha e o algoritmo FIPP-Flex-twofailure provê proteção contra ocorrência de duas falhas simultâneas. Os algoritmos promovem compartilhamento de caminhos de proteção com o maior número de caminhos primários possível.

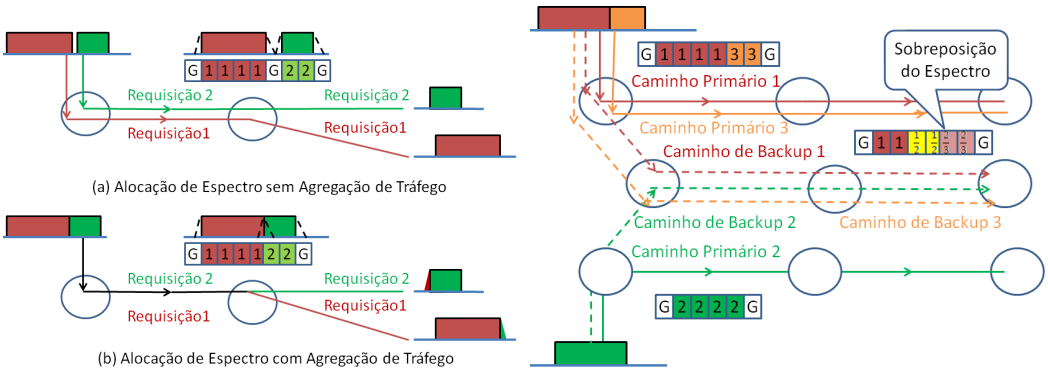
Em [Aibin and Walkowiak 2015], apresenta-se um algoritmo adaptativo para proteção em redes ópticas elásticas. O algoritmo provê sobrevivência de redes ópticas elásticas utilizando multi-enlace e capacidade de alteração do formato de modulação.

Em [Chen et al. 2015], estuda-se o problema do fornecimento de requisições estáticas e dinâmicas com p-cycles FIPP. Para tal, formulou-se um problema de programação linear inteira.

Nenhum dos trabalhos mencionados anteriormente emprega p-cycle FIPP para proteção de caminhos combinado com a técnica de sobreposição do espectro e agregação de tráfego para redes óptica elásticas.

3. Agregação de Tráfego em EONs

Em redes ópticas, agregação de tráfego é uma tecnologia para combinar várias conexões em um caminho óptico sem inserir bandas de guarda entre elas [Zhang et al. 2012]. Em EONs, agregação de tráfego utiliza com mais eficiência os recursos de espectro e reduz o uso de transmissor. Agregação de tráfego pode necessitar de conversão óptico-eléctrico-óptica (OEO) para separar conexões, o que induz o consumo de energia extra e tempo de atraso. Recentemente, um novo esquema de agregação de tráfego tem sido proposto em EON-baseada em OFDM [Zhang et al. 2012]. Este esquema agrega requisições de tráfego originados no mesmo nó de origem e, em seguida, separa-os opticamente, em nós intermediários usando comprimento de onda seletivo e comutadores de largura de banda variável sem causar interferência.



1.1 Agregação de tráfego em EONs 1.2 Sobreposição de espectro

Figura 1. Ilustração

A Figura 1.1 (a) ilustra um exemplo de agregação de tráfego em EONs. Na figura, as requisições 1 e 2 são originadas no mesmo nó fonte. Na agregação de tráfego, reduz-se o uso de banda de guarda como ilustrado na Figura 1.1(b).

No estudo de sobrevivência em rede ópticas quando qualquer enlace de fibra no caminho de trabalho falhar, a conexão é redirecionada em um caminho de backup. Na forma tradicional de compartilhamento de backup, um único caminho de backup pode proteger mais de um caminho primário, desde que sejam totalmente disjuntos. A elasticidade dos transponders ópticos oferece uma nova oportunidade de compartilhamento de espectro (sobreposição de espectro). Dois caminhos de backup podem sobrepor o espectro se os caminhos de trabalho das duas conexões são enlaces disjuntos, e se seus caminhos de backup percorrerem dois caminhos ópticos adjacentes em um enlace de fibra. O esquema de agregação de tráfego que permite a sobreposição do espectro em redes ópticas elásticas diminui a alocação de recursos pelos caminhos de backup, visto que um espectro pode servir a mais de um caminho de backup [Liu et al. 2013].

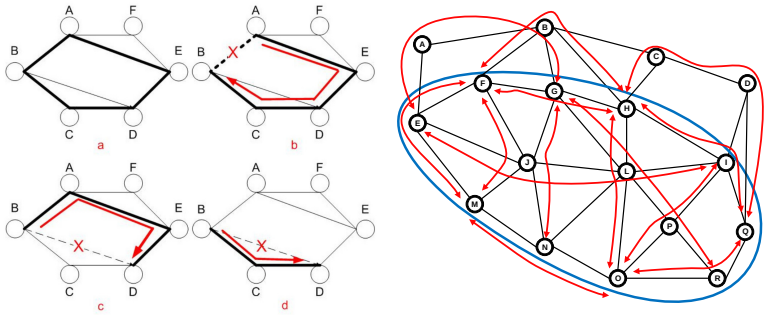
A Figura 1.2 ilustra um exemplo de sobreposição de espectro. Na figura, as requisições 1, 2 e 3 possuem tanto o caminho primário quanto de backup. Na figura, a técnica de agregação de tráfego é mostrada tanto no caminho primário quanto de backup. No exemplo, o caminho primário da requisição 1 é disjunto do caminho primário da requisição 2. Em caso de falha em um enlace apenas uma das conexões fará uso do espectro sobreposto, tornando possível a sobreposição de espectro entre os caminhos de backup destas requisições. Em outro exemplo, o caminho primário da requisição 2 é disjunto do caminho primário da requisição 3, em caso de um enlace falho apenas uma das conexões fará uso do espectro sobreposto, tornando possível a sobreposição de espectro entre os caminhos de backup destas requisições. A figura mostra dois espectros sobrepostos nos caminhos de backup das requisições 1 e 2, e dois espectros se sobrepondo nos caminhos de backup das requisições 2 e 3.

4. p-Cycle

O *p-cycle* é um esquema de proteção em que a capacidade reservada forma estruturas em anel para proteção de redes em malha [Asthana et al. 2010]. *p-Cycles* fornecem proteção semelhante à proteção fornecida por Anéis de Comutação de Linhas Bidirecionais (BLSR), que é considerado uma generalização do esquema de proteção 1:1 [Kiaei et al. 2009]. A diferença fundamental entre p-cycle e proteção em anel é a proteção de enlaces transzonais, que são enlaces que não estão no anel (ciclo) porém os dois nós finais estão. Esta propriedade melhora a eficiência de proteção. Outra propriedade importante é a ausência da necessidade de caminhos que compoem rotas em anel, a fim de serem protegidos. *p-Cycles* provêm rápida restauração pois são pré-conectados [Schupke 2006].

A Figura 2.1 ilustra o conceito de *p-cycle*. Na figura, uma linha simples representa um enlace, um ciclo em negrito representa o *p-cycle* e a seta representa o caminho de recuperação de uma falha denotada por um "X". Na figura 2.1(a), A-B-C-D-E-A é um *p-cycle* que usa a capacidade reservada nos enlaces de proteção. Quando o enlace A-B falha, a proteção é provisionada como ilustrado na figura 2.1(b). Quando o enlace transzonal B-D falha, cada p-cycle proporciona dois caminhos alternativos como mostram as figuras 2.1 (c) e 2.1(d).

Um caso especial de *p-cycle* para proteção de caminho é o chamado *p-cycle* de Proteção de Caminho com Independência de Falha (FIPP) [Kodian and Grover 2005]. *p-Cycles* FIPP fornecem proteção de caminhos para caminhos com nós finais sobre o *p-cycle*. FIPP é uma extensão do conceito de *p-cycle* que permite que falhas não se-



2.1 Tradicional 2.2 FIPP

Figura 2. Exemplos de p-cycle

jam necessariamente limitadas a um enlace ou segmento de caminho imediatamente adjacente aos nós terminais. FIPPs baseiam-se em diferentes "caminhos de trabalho" e de "*backup*", e proporciona a vantagem da detecção de falha ser independente da localização da falha, ou seja, ser "independente de falha". Tal propriedade é vantajosa quando a localização da falha é lenta ou difícil, como em redes transparentes e translúcidas [Kodian and Grover 2005].

A Figura 2.2 ilustra o conceito de *p-cycle* FIPP. Na Figura 2.2, o ciclo em destaque EFGHIQRONM representa o p-cycle e as setas mostram os vários caminhos que são protegidos. Nesses caminhos, estão tanto enlaces transzonais quanto enlaces do próprio p-cycle. Pode-se observar que um único p-cycle protege um conjunto de catorze caminhos mutuamente excludentes e que possuem seus nós de extremidade no p-cycle de caminho. Nota-se que a proteção através de p-cycle FIPP possui uma eficiência na economia de recurso comparada com outras técnicas de proteção.

A proteção por compartilhamento de caminhos de reserva (SBPP) propostas para redes baseadas em sinalização IP também tem a propriedade de falhas independentes. No entanto, a principal diferença entre SBPP e FIPP é que em SBPP o caminho de *backup* precisa ser determinado em tempo real, o que pode levar a adoção de um caminho de restauração sem a qualidade de transmissão adequada, em caso de falha. Por outro lado em FIPPs, caminhos de proteção pré-conectado são muito importantes para assegurar o nível de proteção desejado. Além disso, SBPP exige uma extensa base de dados, devido à necessidade de todos os nós terem conhecimento da capacidade global, topologia e compartilhamento de caminho de *backup*.

5. O Algoritmo RSA-Flex

Neste trabalho, utiliza-se o algoritmo RSA-Flex [Oliveira and da Fonseca 2014], para o roteamento e alocação de espectro que será descrito a seguir.

O problema de roteamento e alocação de espectro é um problema NP-difícil [Wang et al. 2011] e heurísticas são necessárias para resolver o problema. No problema de RSA além da restrição de continuidade de espectro que impõe a utilização do mesmo espectro em cada fibra ao longo da rota de um caminho óptico, *slots* devem ser alocados de forma contínua no espectro, restrição denominada contiguidade do espectro. Como em [Oliveira and da Fonseca 2014] os algoritmos no presente modelam a disponibilidade de espectro na rede como um multigrafo rotulado. Um multigrafo é um grafo que pode ter

várias arestas (também chamadas de "arestas paralelas"), conectando o mesmo par de nós. Neste grafo, os vértices representam OXCs e as arestas representam os *slots* dos enlace que conectam os OXCs. Nos multigrafos utilizados nesse artigo, todos os vértices são conectados por N arestas, que é o número de *slots* no espectro de cada enlace da rede. O $W_{s,d}$ representa a disponibilidade de *slots*. O valor ∞ significa que o *slot* já está alocado enquanto o valor 1 significa que o *slot* está disponível para alocação. Estes valores foram definidos para facilitar o emprego de algoritmos tradicionais que encontram o caminho mais curto.

No procedimento proposto, o multigrafo é transformado em $N - b + 1$ grafos, sendo b a demanda de largura de banda em *slots* requisitada. Estes grafos são gerados através da seleção de uma aresta no multigrafo e das b arestas consecutivas a aresta selecionada. Este conjunto de b arestas do multigrafo é mapeado em uma única aresta do grafo gerado. Seu peso é dado pela aplicação de uma função de peso específica, que considera o peso das b arestas. A Figura 3 ilustra um multigrafo representando o espectro e um dos grafos gerado. Para cada grafo gerado, executa-se um algoritmo de caminho mais curto e o caminho escolhido é o que tem o menor peso entre todos os caminhos mais curtos encontrados.

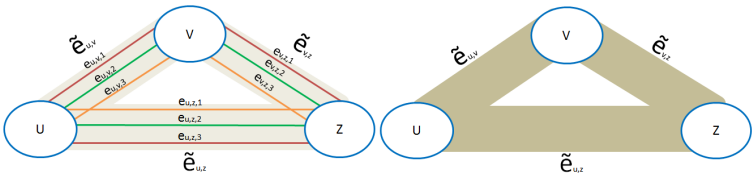


Figura 3. Multigrafo em um grafo associado

Para uma demanda de b *slots*, $N - b + 1$ grafos do tipo $\widetilde{G}_{n,b}$ são gerados. Cada aresta do grafo $\widetilde{G}_{n,b}$ corresponde ao mapeamento de b arestas de G iniciando na n-ésima aresta de G. Uma vez que as mesmas arestas ordenadas conectando dois nós em G são mapeadas em arestas de $\widetilde{G}_{n,b}$, a continuidade do espectro é garantida.

A fim de facilitar a compreensão dos algoritmos, introduz-se a notação usada no artigo. Seja:

s: o nó fonte; d: o nó destino;

b: a demanda de largura de banda nos *slots*, $b = 1 \ldots B$;

N: número de *slots* entre dois nós;

$r(s, d, b)$: requisição do nó s para o nó d com demanda de largura de banda b em *slots*;

$G = (V, E, W)$: multigrafo marcado composto por um conjunto de nós V, um conjunto de arestas E e um conjunto de pesos das arestas W, $|E| = N \cdot |V|$.

$G' = (V, E, W)$: multigrafo marcado que considera uma aresta que está sendo utilizada por um caminho de proteção como disponível, desde que o caminho protegido pelo caminho de proteção seja disjunto do novo caminho a ser criado. O multigrafo é composto por um conjunto de nós V, um conjunto de arestas E e um conjunto de pesos das arestas W, $|E| = N \cdot |V|$.

$\widetilde{V} = V$: conjunto de nós; $E = \{e_{u,v,n}\}$: conjunto de n arestas;

$e_{u,v,n}$: a n-ésima arestas conectando os nós u e v; $w(e_{u,v,n})$: o peso das arestas $e_{u,v,n}$;

$w(e_{u,v,n}) = 1$ se o n-ésimo *slot* no enlace conectando os nós u e v estão livres e $w(e_{u,v,n}) = \infty$ se o *slot* está alocado;

$W = \{w(e_{u,v,n})\}$:conjunto de pesos das arestas;

$\widetilde{G}_{n,b} = (\widetilde{V}, \widetilde{E}, \widetilde{C})$: O n-ésimo grafo marcado tal que $\widetilde{E}$ é o conjunto de arestas conectando $\{\widetilde{u}, \widetilde{v}\} \in \widetilde{V}$ e $\widetilde{W}$ é o conjunto de custos associados a $\widetilde{E}$. As arestas em $\widetilde{E}$ correspondem ao mapeamento de b arestas de G, sendo a primeira aresta a n-ésima aresta conectando u e v.

$\widetilde{e}_{u,v} \in \widetilde{E}$: arestas conectando $\widetilde{u}$ e $\widetilde{v}$; $\widetilde{e}_{\widetilde{u,v}} = \{e_{u,v,n}\} \in E$ é uma sequência tal que $e_{u,v,n}$ é a menor aresta ordenada, $e_{u,v,n+b}$ é a maior aresta ordenada e $|\widetilde{e}_{u,v}| = b$;

$\widetilde{w}_n(\widetilde{e}_{\widetilde{u,v}})$: peso da aresta $\widetilde{e}_{\widetilde{u,v}}$;

$\widetilde{W}_n = \{\widetilde{w}_n(\widetilde{e}_{\widetilde{u,v}})\}$: conjunto de pesos de arestas;

P_n: sequência de arestas $\widetilde{G}_n$ tal que o nó fonte s é o menor nó ordenado e d é o maior nó ordenado;

$W(\widetilde{P}_n)$: $\sum_{\widetilde{e}_{\widetilde{u,v}} \in \{\widetilde{P}_n\}} \widetilde{e}_{\widetilde{u,v}}$: o peso do caminho $\widetilde{P}_n$ é a soma dos pesos de todos as arestas na cadeia;

$W_{s,d}$ = peso do menor caminho entre s e d;

$\widetilde{c}_{u,v,b}$: *p-cycle* que contém os vértices u e v e arestas correspondentes ao mapeamento de b arestas do multigrafo G;

$\widetilde{C}_{u,v,b} = \widetilde{c}_{u,v,b}$: conjunto de todos os *p-cycles* que contém os vértices u e v e arestas correspondentes ao mapeamento de b arestas do multigrafo G;

$\widetilde{C}$: conjunto de todos os *p-cycles* estabelecidos;

$P_1 \oplus P_2$: concatenação de dois caminhos P_1 e P_2 disjuntos

Algorithm 1 RSA-Flex

1: $\forall n = 1...N-b$
2: $(W(P_n), P_n) = ShortestPath(\widetilde{G}_{n,b}, r(s,d,b))$
3: $W_{s,d} = W(P_n) | \, \forall i \, W(P_n) \leq W(P_i)$
4: **if** $W_{s,d} = \infty$ **then**
5: $block \; r(s,d,b)$
6: **else**
7: $W(e_{u,v,i}) = \infty \quad \forall \{u,v\} \in \widetilde{P}_i \quad n = n...i+b-1$
8: **end if**

No algoritmo 1, a linha 1 estabelece o conjunto de arestas que será mapeada para $\widetilde{G}_{n,b}$ arestas. A linha 2 resolve um algoritmo de menor caminho para o grafo $\widetilde{G}_{n,b}$ e provê o menor caminho e seu respectivo peso. Se o peso do caminho mais curto for ∞, não é possível encontrar um caminho com a restrição de continuidade para a demanda b com alocação começando com o n-ésimo *slot*. A linha 3 seleciona o caminho entre os $N-b+1$ caminhos mais curtos, com o menor peso. No caso do peso de todos os caminhos mais curtos serem ∞ (linha 4), não existe caminho na rede que satisfaça a requisição de b *slots*, portanto, a requisição é bloqueada (Linha 5). Caso contrário, o caminho mais curto com o menor peso será escolhido (linha 7) e as arestas correspondentes no multigrafo G terão seu peso alterado para ∞ (linha 8), o que significa que os *slots* são alocados para o caminho óptico recém-estabelecido.

Uma vez que o algoritmo RSA-Flex executa um algoritmo de caminho mais curto

$N - b$ vezes e considerando o uso do algoritmo de caminho mais curto de Dijkstra, a complexidade computacional do algoritmo proposto é $N \cdot (|V| + |E|) \cdot \log(|V|)$.

6. Algoritmo FIPPSh-Flex

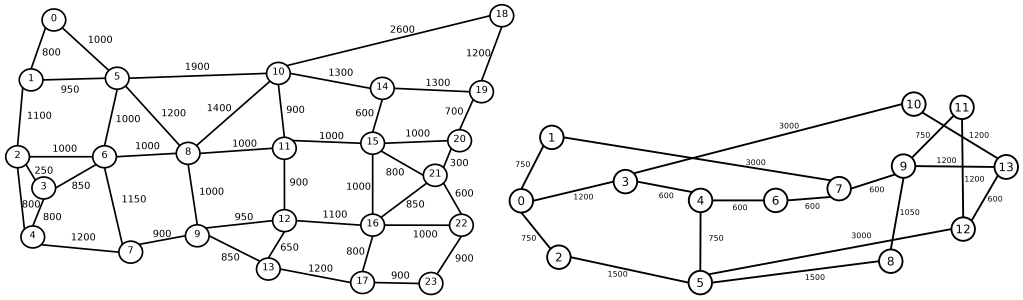
O algoritmo FIPP-Flex, [Oliveira and da Fonseca 2014], resolve o estabelecimento de caminhos ópticos em redes protegidas por *p-cycles* FIPP. O algoritmo FIPPSh-Flex, resolve o estabelecimento de caminhos ópticos em redes protegidas por *p-cycles* FIPP e estende algoritmo FIPP-Flex para permitir agregação de tráfego em seus caminhos ópticos bem como sobreposição do espectro entre caminhos de backup disjuntos (seção 3). A sobreposição do espectro é realizada no espectro de caminhos de backup que protegem caminhos primários adjacentes. Como o algoritmo FIPP-Flex gera uma melhor utilização dos recursos da rede que outros algoritmos existentes [Oliveira and da Fonseca 2014], o uso de agregação reduz ainda mais o uso dos recursos da rede. A melhor utilização dos recursos no algoritmo FIPPSh-Flex é possível devido também ao fato do caminho de backup permitir o estabelecimento de outros caminhos utilizando slots já ocupados por outros caminhos de backup.

Algorithm 2 FIPPSh-Flex

1: $(W(P_n), P_n) = RSA - Flex(G, s, d, b)$
2: **if** $W_{s,d} = \infty$ **then**
3: $block\ r(s, d, b)$
4: **else**
5: **if** $C_{u,v,i} \neq \emptyset \forall i \geq b$ **then**
6: establish $r(s, d, b)$ as P_n
7: **else**
8: $(W(P_1), P_1)$ = RSA-Flex $(G', r(s, d, b))$
9: $(W(P_2), P_2)$ = RSA-Flex $(G', r(s, d, b))$
10: **if** $W(P_1) = \infty$ or $W(P_2) = \infty$ **then**
11: block $r(s, d, b)$
12: **else**
13: establish $r(s, d, b)$ as P_n
14: establish P_1 and P_2
15: $\widetilde{c}_{u,v,b} = P_1 \oplus P_2$
16: **end if**
17: **end if**
18: **end if**

No algoritmo FIPPSh-Flex, caminhos ópticos são estabelecidos se, e somente se, puderem ser protegidos por um *p-cycle* FIPP, que protege somente caminhos primários disjuntos. O algoritmo FIPPSh-Flex garante um caminho de proteção para cada caminho óptico estabelecido e a proteção é garantida para falhas individuais.

No algoritmo 2, a linha 1 tenta encontrar um caminho para estabelecer a requisição $r(s, d, b)$. Se não houver caminho disponível, então o pedido é bloqueado (linha 3). Caso exista, procura-se um *p-cycle* para proteger o caminho solicitado (linha 5). Se existir um *p-cycle*, o caminho óptico é estabelecido. Caso contrário, tenta-se descobrir um *p-cycle* em potencial para proteger o caminho óptico solicitado (linhas 8 e 9). A principal diferença entre os algoritmos FIPP-Flex e FIPPSh-Flex está nas linhas 8 e 9, pois no algoritmo FIPPSh-Flex estas linhas consideram a agregação de tráfego e sobreposição de espectro na criação do p-cycle, provendo uma melhor eficiência dos recurso de backup utilizados. Se não existir nenhum *p-cycle* que possa ser criado para proteger o caminho óptico, a requisição é bloqueada (linha 11); caso contrário, o caminho óptico (linha 13), bem como o *p-cycle* (linhas 14 e 15) são estabelecidos para satisfazer a requisição.



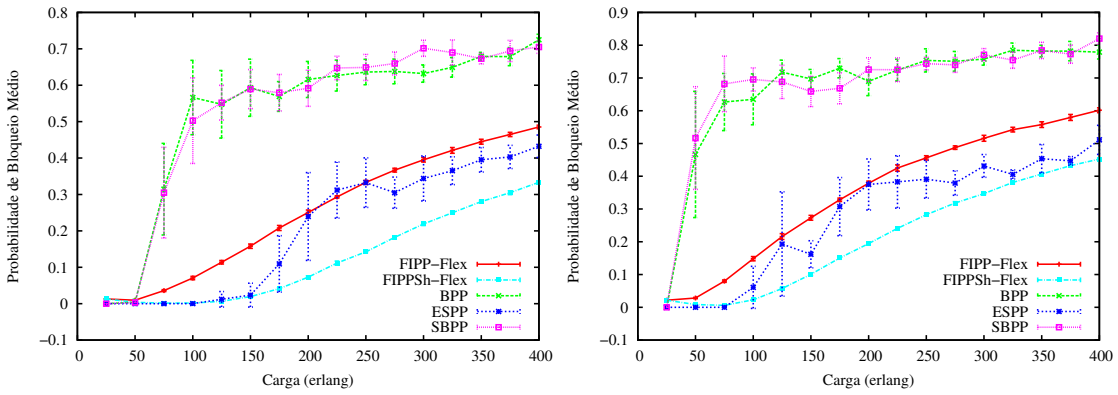
4.1 USA 4.2 NSF

Figura 4. Topologias

7. Avaliação de Desempenho

Para avaliar o desempenho do algoritmo FIPPSh-Flex, foram empregados experimentos de simulação e os resultados comparados com os esquemas de proteção FIPP-Flex, SBPP, ESPP e BPP. O simulador *FlexGridSim* [Moura and Drummond] foi utilizado nas simulações. Em cada simulação, 100.000 requisições foram geradas. Utilizou-se o método de replicação independente e adotou-se nível de confiança de 95% para os intervalos de confiança. As topologias NSF (Figura 4.1) e USA (Figura 4.2) foram utilizadas, as figuras mostram as respectivas distancias entre os nós. Na rede elástica simulada, a análise do espectro foi dividido em 240 *slots* de 12,5 GHz cada.

Nas próximas figuras, curvas rotulada como BPP mostram resultados para redes protegidas pelo esquema de proteção 1+1, curvas rotulada como SBPP para redes protegidas pelo algoritmo proposto em [Shao et al. 2012], curvas rotuladas como FIPP-Flex mostram resultados para redes protegidas por p-cycle criado pelo algoritmo FIPP-Flex e curvas rotulada como ESPP para redes protegidas pelo algoritmo proposto em [Liu et al. 2013] que utiliza o método de agregação de tráfego e sobreposição de espectro na criação caminhos ópticos de backup compartilhados. Finalmente curvas rotulada como FIPPSh-Flex mostram resultados para redes protegidas pelo algoritmo proposto (FIPPSh-Flex) que realiza a proteção de caminhos primários através de *p-cycles* criados utilizando o método de agregação de tráfego e sobreposição de espectro.



5.1 USA 5.2 NSF

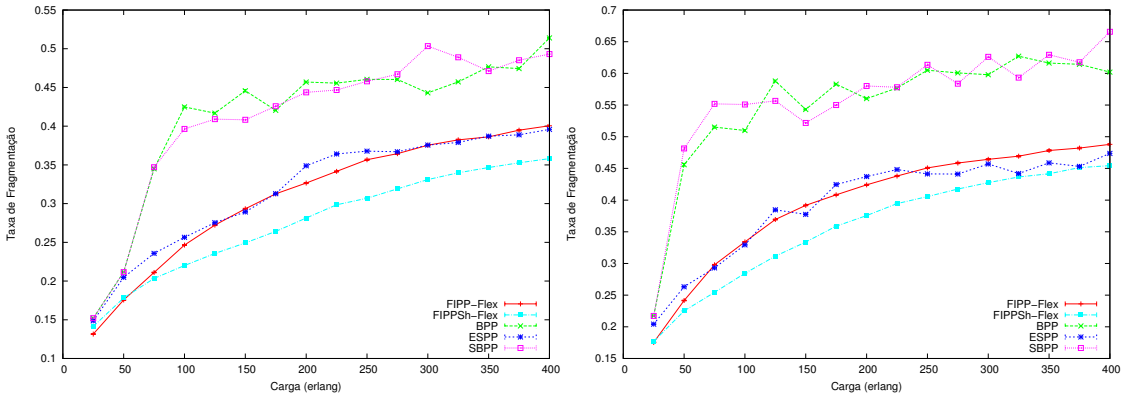
Figura 5. Bloqueio de banda em função da carga da rede

As Figuras 5.1 e 5.2 mostram a probabilidade de bloqueio (*Bandwidth Blocking*

Ratio) para a topologia USA e NSF, respectivamente. Os algoritmos SBPP e BPP produzem comportamento de BBR semelhantes em ambas topologias.

Para a topologia USA (Figura 5.1) os algoritmos BPP e SBPP saturam a rede com carga de 100 erlangs. Devido à alta conectividade da topologia USA não há bloqueio até 50 erlangs. O FIPP-Flex gera 50% menos bloqueio do que BPP e SBBP, isto ocorre devido à propriedade de compartilhamento do p-cycle FIPP. Por sua vez ESPP produz aproximadamente 33% menos bloqueio que FIPP-Flex entre as cargas 50 e 200 erlangs e entre as cargas 250 e 400 erlangs, apesar do algoritmo FIPP-Flex não utilizar as propriedades de agregação de tráfego e sobreposição de espectro a propriedade de compartilhamento do p-cycle gera baixo bloqueio, produzindo bloqueio bem próximo ao gerado pelo algoritmo ESPP. Até a carga de 150 erlangs FIPPSh-Flex e ESPP praticamente não geram bloqueio, consequência do uso de sobreposição de espectro e agregação de tráfego. O algoritmo FIPPSh-Flex combina as vantagens do compartilhamento de p-cycle com agregação de tráfego e sobreposição de espectro, isso faz com que FIPPSh-Flex gere menos bloqueio que os outros algoritmos. O FIPPSh-Flex produz 30% menos bloqueio que o ESPP a partir de 150 erlangs, evidenciando a vantagem de se usar p-cycle já que ambos os algoritmos utilizam agregação de tráfego e sobreposição de espectro .

Para a topologia NSF (Figura 5.2) os algoritmos BPP e SBPP saturam a rede com carga de 75 erlangs. O FIPP-Flex gera 42% menos bloqueio que o BPP e SBBP, isto ocorre devido à propriedade de compartilhamento do p-cycle FIPP. Por sua vez, ESPP gera aproximadamente 50% menos bloqueio que o FIPP-Flex entre as cargas 50 e 125 erlangs e entre as cargas 200 e 400 erlangs. O FIPP-Flex produz baixo bloqueio, bem próximo ao gerado pelo ESPP. Até a carga de 75 erlangs os algoritmos FIPPSh-Flex e ESPP praticamente não geram bloqueio. O algoritmo FIPPSh-Flex produz 15% menos bloqueio que o algoritmo ESPP a partir de 75 erlangs. As propriedades de sobreposição de espectro e agregação de tráfego dos algoritmos ESPP e FIPPSh-Flex torna-se visivelmente vantajosas, em ambas topologias, quando comparado o comportamento destes algoritmos com os algoritmos BPP, SBPP e FIPP-Flex que não utilizam tais propriedades.



6.1 USA 6.2 NSF

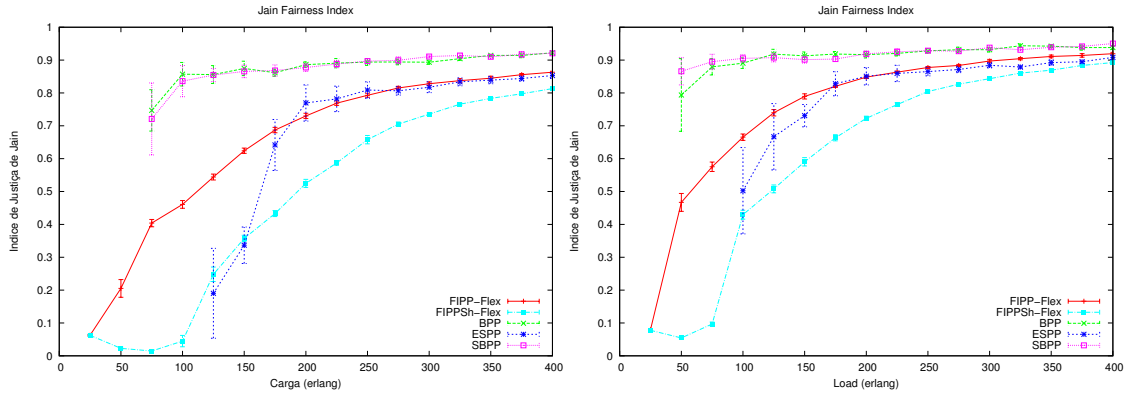
Figura 6. Relação de fragmentação

As Figuras 6.1 e 6.2 representam a relação de fragmentação em função da carga da rede para as topologia USA e NSF, respectivamente. Em redes óptica elásticas, o estabelecimento e desestabelecimento de caminhos ópticos leva à fragmentação do espectro que

é um estado em que há *slots* disponíveis, mas não podem ser reunidos de uma maneira que possam ser usado para aceitar novas requisições. A proporção de fragmentação é definida como a razão média entre o número de tipos de exigências que não podem ser aceitas e o número total de tipos de demandas. Para ambas as topologias, o comportamento de fragmentação é bem similar e influenciam a BBR gerada.

Para a topologia USA (Figura 6.1), o algoritmo SBPP produz taxa de fragmentação 5% menor que o algoritmo BPP, isto ocorre devido ao compartilhamento de caminhos realizado pelo algoritmo SBPP. O FIPP-Flex gera taxa de fragmentação 23% menor que SBPP, isto ocorre pois o p-cycle pode ser compartilhado entre qualquer nó presente no p-cycle, já o caminho de backup só pode ser compartilhado entre nós finais. O FIPP-Flex produz taxa de fragmentação 6% menor que o ESPP apesar do algoritmo ESPP possuir menor bloqueio que o algoritmo FIPP-Flex, isto ocorre pois o p-cycle possui maior compartilhamento, gerando uma quantidade menor de caminhos de backup desestabelecido, diminuindo o número de espectros disponíveis que não pode ser usado. O algoritmo FIPPSh-Flex produz taxa de fragmentação 17% menor que o algoritmo ESPP.

Para a topologia NSF (Figura 6.2), o algoritmo SBPP gera taxa de fragmentação 3% menor que o algoritmo BPP, isto ocorre devido ao compartilhamento de caminhos presente no algoritmo SBPP. O algoritmo FIPP-Flex produz taxa de fragmentação 18% menor que o algoritmo SBPP, isto ocorre pois o p-cycle pode ser compartilhado entre qualquer nó presente no p-cycle, já o caminho de backup só pode ser compartilhado entre nós finais. O algoritmo FIPP-Flex gera taxa de fragmentação 3% menor que o algoritmo ESPP apesar do algoritmo ESPP gerar menor bloqueio que o algoritmo FIPP-Flex, isto ocorre pois como o p-cycle possui maior compartilhamento entre os nós, o número de caminhos de backup desestabelecido é menor, influenciando a fragmentação. O algoritmo FIPPSh-Flex produz taxa de fragmentação 20% menor que o algoritmo ESPP . Os algoritmos BPP e SBPP possuem alta taxa de fragmentação isto é consequência do bloqueio gerado e do baixo número de caminhos alternativos.



7.1 USA 7.2 NSF

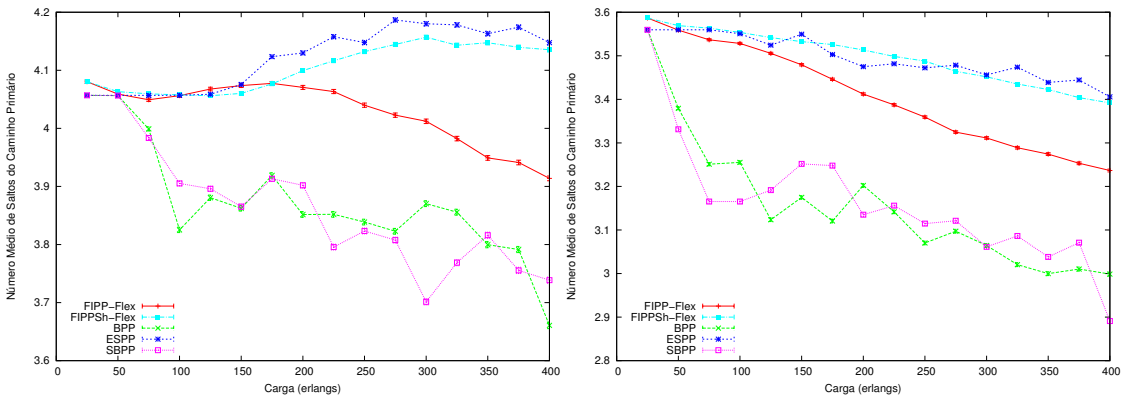
Figura 7. Índice de justiça de Jain

As Figuras 7.1 e 7.2 mostram o índice de justiça de Jain (JFI) usado por diferentes pares origem - destino, nas topologia USA e NSF, respectivamente. Os algoritmos BPP e SBPP possuem altos valores de índice Jain, distribuindo o bloqueio das solicitações de maneira mais uniforme entre os pares fonte e destino. Isso acontece pois estes algoritmos

produzem maior BBR. Os algoritmos FIPP-Flex, FIPPSh-Flex e ESPP possuem baixo índice Jain devido à baixa probabilidade de bloqueio que estes possuem, especialmente sob cargas baixas.

Para a topologia USA (Figura 7.1) o algoritmo o FIPP-Flex gera índice de justiça de Jain 13% menor que o SBPP, isto ocorre devido ao maior bloqueio produzido pelo algoritmo SBPP. O ESPP produz índice de justiça de Jain 60% menor que o FIPP-Flex até carga de 175 erlangs, isto ocorre devido às propriedades de compartilhamento de espectro e agregação de tráfego do ESPP. A partir de 150 erlangs o algoritmo FIPPSh-Flex produz índice de justiça de Jain 16% menor que o algoritmo ESPP, isto ocorre devido ao maior bloqueio produzido pelo algoritmo ESPP e o compartilhamento do p-cycle FIPP.

Para a topologia NSF (Figura 7.2) o algoritmo FIPP-Flex produz índice de justiça de Jain 14% menor que o algoritmo SBPP, isto ocorre devido ao maior bloqueio produzido pelo algoritmo SBPP. O algoritmo ESPP gera índice de justiça de Jain 30% menor que o algoritmo FIPP-Flex, isto ocorre devido às propriedades de compartilhamento de espectro e agregação de tráfego do algoritmo ESPP. O algoritmo FIPPSh-Flex gera índice de justiça de Jain 24% menor que o algoritmo ESPP, isto ocorre devido ao maior bloqueio produzido pelo algoritmo ESPP e o compartilhamento do p-cycle FIPP.



8.1 USA 8.2 NSF

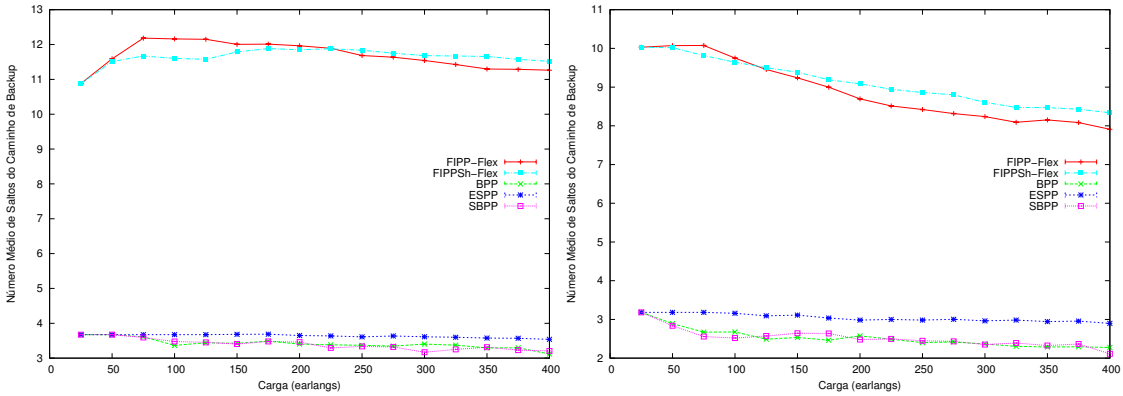
Figura 8. Número médio de saltos do caminho primário

As Figuras 8.1 e 8.2 mostram o número médio de saltos para o caminho primário para as topologia USA e NSF, respectivamente. Para ambas as topologias os algoritmos BPP e SBPP possuem baixos número médio de saltos, principalmente para cargas altas, ao contrário dos algoritmos FIPP-FLex, FIPPSh-Flex e ESPP.

Para a topologia USA (Figura 8.1) nota-se que com o aumento da carga, os algoritmos BPP, SBPP e FIPP-Flex diminuem o número médio de saltos no caminho primário, indicando a diminuição da disponibilidade dos enlaces que causam um maior bloqueio. Por sua vez os algoritmos ESPP e FIPPSh-Flex, que utilizam sobreposição de espectro e agregação de tráfego aumentam o número de saltos demonstrando que a rede mesmo sobrecarregada consegue prover caminhos maiores para as requisições.

Para a topologia NSF (Figura 8.2) que possui baixa conectividade, com o aumento da carga todos os algoritmos diminuem o número de saltos no caminho primário. No entanto, os algoritmos ESPP e FIPPSh-Flex possuem uma menor queda no número de

saltos, o que indica uma menor sobrecarga na rede em relação aos outros algoritmos.



9.1 USA 9.2 NSF

Figura 9. Número médio de saltos do caminho de backup

As Figuras 9.1 e 9.2 comparam o número médio de saltos para o caminho de backup nas topologia USA e NSF, respectivamente.

Para a topologia USA (Figura 9.1) a necessidade de criação de anéis virtuais para proteção nos algoritmos FIPP-Flex e FIPPSh-Flex geram caminhos com três vezes mais saltos que os algoritmos BPP, SBPP e ESPP. Como p-cycles FIPP permitem ser compartilhados entre qualquer nó presente no p-cycle, o alto número de saltos não gera sobrecarga na rede nem aumento do bloqueio da rede.

Para a topologia NSF (Figura 9.2), os algoritmos FIPP-Flex e FIPPSh-Flex geram caminhos com o dobro de saltos dos algoritmos BPP, SBPP e ESPP. A diferença de saltos nas topologias USA e NSF para os algoritmos FIPP-Flex e FIPPSh-Flex é causado devido à menor conectividade da rede. Em ambas as topologias apesar dos algoritmos que utilizam *p-cycle* necessitarem de um caminho maior, a alta capacidade de compartilhamento consegue prover proteção para um maior número de caminhos, diferentemente das outras soluções, não aumentando a probabilidade de bloqueio.

8. Conclusão

Este artigo introduziu um algoritmo para suportar o estabelecimento de caminhos ópticos em redes ópticas elásticas protegidas por p-cycle utilizando agregação de tráfego e sobreposição de espectro. A técnica p-cycle realiza a restauração rápida de proteção em forma de anel e eficiente capacidade de proteção em malha. O algoritmo foi avaliado para diferentes topologias e cargas. Os algoritmos que utilizam sobreposição de espectro mostram resultados mais atraentes para ambas as topologias em comparação com os outros algoritmos. O algoritmo FIPPSh-Flex produz até 20% menos bloqueio que a técnica ESPP, que também utiliza sobreposição de espectro e agregação de tráfego. A capacidade de sobreposição de espectro é visivelmente eficiente, principalmente quando combinada com o p-cycle FIPP.

9. Agradecimento

O presente trabalho foi realizado com apoio do CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil, processo número 165446/2015-3 e INCT

Fotonicom.

Referências

Aibin, M. and Walkowiak, K. (2015). Adaptive survivability algorithm for path protection with various traffic classes in elastic optical networks. In *Reliable Networks Design and Modeling (RNDM), 2015 7th International Workshop on*, pages 56–62.

Asthana, R., Singh, Y., and Grover, W. (2010). p-cycles: An overview. *IEEE, Communications Surveys Tutorials*, 12(1):97 –111.

Chen, X., Zhu, S., Jiang, L., and Zhu, Z. (2015). On spectrum efficient failure-independent path protection p-cycle design in elastic optical networks. *Lightwave Technology, Journal of*, 33(17):3719–3729.

Din, D.-R. and Lai, I.-R. (2015). Multicast protection problem on elastic optical networks using segment-base protection. In *Informatics, Electronics Vision (ICIEV), 2015 International Conference on*, pages 1–6.

Jinno, M., Takara, H., Kozicki, B., Tsukishima, Y., Sone, Y., and Matsuoka, S. (2009). Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. *Communications Magazine, IEEE*, 47(11):66–73.

Kiaei, M., Assi, C., and Jaumard, B. (2009). A survey on the p-cycle protection method. *IEEE, Communications Surveys Tutorials*, 11(3):53 –70.

Kodian, A. and Grover, W. (2005). Failure-independent path-protecting pcycles: efficient and simple fully preconnected optimal-path protection. *IEEE, J. Lightwave Technol.*

Liu, M., Tornatore, M., and Mukherjee, B. (2013). Survivable traffic grooming in elastic optical networks -shared protection. *Lightwave Technology, Journal of*, 31(6):6230–6234.

Moura, P. M. and Drummond, A. C. FlexGridSim: Flexible Grid Optical Network Simulator. http://www.lrc.ic.unicamp.br/FlexGridSim/.

Oliveira, H. and da Fonseca, N. (2014). Protection in elastic optical networks against up to two failures based fipp p-cycle. In *Computer Networks and Distributed Systems (SBRC), 2014 Brazilian Symposium on*, pages 369–375.

Schupke (2006). Analysis of p-cycle capacity in WDM networks. *Photonic Network Communications*, pages p. 41–51.

Shao, X., Yeo, Y.-K., Xu, Z., Cheng, X., and Zhou, L. (2012). Shared-path protection in OFDM-based optical networks with elastic bandwidth allocation. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference*, pages 1–3.

Wang, Y., Cao, X., and Pan, Y. (2011). A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks. In *Proc of IEEE,INFOCOM 2011*, pages 1503–1511.

Zhang, G., De Leenheer, M., and Mukherjee, B. (2012). Optical traffic grooming in ofdm-based elastic optical networks [invited]. *Optical Communications and Networking, IEEE/OSA Journal of*, 4(11):B17–B25.

Trilha Principal do SBRC 2016
Sessão Técnica 9
Redes Centradas na Informação

Aqui Todos Podem Publicar: Uma Abordagem Escalável para Controle de Acesso Muitos para Muitos em Redes Centradas em Informação

Rafael Hansen da Silva, Weverton Luis da Costa Cordeiro, Luciano Paschoal Gaspary

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

Resumo. *Um dos principais desafios em Redes Centradas em Informação (ICN) é como prover controle de acesso à publicação e recuperação de conteúdos. Apesar das potencialidades, as soluções existentes geralmente consideram um único usuário agindo como publicador. Ao lidar com múltiplos publicadores, elas podem levar a uma explosão combinatória de chaves criptográficas. As soluções projetadas visando aos múltiplos publicadores, porém, dependem de arquiteturas de rede específicas e/ou de mudanças na mesma para operar. Neste artigo é proposta uma solução, apoiada em criptografia baseada em atributos, para controle de acesso a conteúdos. Nessa solução, o modelo de segurança é voltado a grupos de compartilhamento seguro, nos quais todos os usuários membros podem publicar e consumir conteúdos. Diferente de trabalhos anteriores, a solução proposta mantém o número de chaves proporcional ao de membros nos grupos e pode ser empregada em qualquer arquitetura ICN de forma gradual. A proposta é avaliada quanto ao custo de operação, à quantidade de chaves necessárias, e à eficiência na disseminação de conteúdos. Em comparação às soluções existentes, ela oferece maior flexibilidade no controle de acesso, sem aumentar a complexidade do gerenciamento de chaves e sem causar sobrecustos significativos à rede.*

Abstract. *One of the main challenges in Information Centric Networks (ICN) is providing access control to content publication and retrieval. In spite of the potentialities, existing solutions often consider a single user acting as publisher. When dealing with multiple publishers, they may lead to a combinatorial explosion of cryptographic keys. Those solutions that focus on multiple publishers, on the other hand, rely on specific network architectures and/or changes to operate. In this paper we propose a solution, supported by attribute-based encryption, for content access control. In this solution, the security model is focused on secure content distribution groups, in which any member user can publish to and retrieve from. Unlike previous work, the proposed solution keeps the number of cryptographic keys proportional to the number of group members, and may even be adopted gradually in any ICN architecture. The proposed solution is evaluated with respect to the overhead it imposes, number of required keys, and efficiency of content dissemination. In contrast to existing solutions, it offers higher access control flexibility, without increasing key management process complexity or causing significant network overhead.*

1. Introdução

O paradigma de Redes Centradas em Informação (*Information Centric Networks, ICN*) emergiu como uma direção promissora para a Internet do Futuro [Xylomenos et al. 2014].

Apesar de suas potencialidades – por exemplo, de tornar a distribuição de conteúdos escalável e eficiente, e diminuir o tráfego no núcleo da Internet [Ahlgren et al. 2012], há, ainda, diversos desafios que precisam ser abordados. Um dos mais importantes, e decisivo para o sucesso desse paradigma, está relacionado ao controle de acesso [de Brito et al. 2012, Xylomenos et al. 2014]. Uma vez que os conteúdos passam a ser recuperados a partir de *caches* distribuídas na rede, os mecanismos de segurança precisam garantir que conteúdos publicados de forma protegida (isto é, com restrições de acesso) sejam consumidos apenas por usuários devidamente autorizados.

As soluções existentes geralmente focam em cenários em que grupos de compartilhamento seguro de conteúdo são formados por um publicador e vários consumidores [Misra et al. 2013, Papanis et al. 2014], sendo interessantes para uso por provedores como YouTube, Google Play, iTunes Store e NetFlix. No entanto, elas podem levar a um problema de explosão combinatória de chaves criptográficas, caso adotadas em cenários em que grupos formados por múltiplos publicadores e consumidores são a norma. As soluções focadas em múltiplos publicadores, no entanto, introduzem entidades extras na rede para realizar reencriptação de conteúdos e/ou controle de acesso [Fotiou et al. 2012, Singh et al. 2012]. Embora efetivas, elas são intrusivas e pouco flexíveis para adoção de forma gradual, além de serem vulneráveis ao comportamento malicioso dessas entidades e, em alguns casos, depententes de arquitetura.

Para lidar com esses problemas, neste artigo, propõe-se um modelo de segurança, apoiado por criptografia baseada em atributos, para controle de acesso a conteúdos em ICN. O modelo utiliza o conceito de participação em grupos de compartilhamento seguro, nos quais apenas usuários membros podem recuperar conteúdos. A publicação pode ser refinada pelo uso de atributos de usuários, de modo a restringir a recuperação a subconjuntos específicos de membros. O modelo proposto concilia suporte a múltiplos publicadores e controle de acesso agnóstico de arquitetura, assim, mantendo o número de chaves proporcional ao de membros nos grupos, e sem depender de entidades centrais. O modelo proposto é avaliado quanto ao suporte a múltiplos publicadores e ao custo de operação. Os resultados alcançados, por meio de avaliações em ambiente experimental controlado, confirmam a efetividade do modelo, o qual introduz um custo marginal para a publicação e a recuperação de conteúdos (em comparação às soluções existentes), ao passo que torna mais robusto e escalável o controle de acesso.

O restante do artigo está organizado como segue. A Seção 2 discute os principais trabalhos relacionados. A Seção 3 descreve em detalhes a solução proposta para controle de acesso a conteúdos em ICN, enquanto que a Seção 4 discute o ambiente experimental utilizado para avaliação da solução e os principais resultados alcançados. Por fim, a Seção 5 conclui o artigo com considerações finais e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

Criptografia é o mecanismo mais fundamental para se implementar a publicação segura e privativa de conteúdos [Jacobson et al. 2012]. No entanto, os mecanismos de criptografia simétrica e assimétrica não são suficientes se empregados isoladamente em ICN: enquanto a primeira requer algum recurso externo (*ex.* telefone ou *e-mail*) para o processo de distribuição de chaves, a segunda torna inócuas as facilidades de *cache* na rede, uma vez que o conteúdo precisa ser cifrado para cada usuário.

De forma geral, as soluções propostas possuem como objetivos em comum o maior aproveitamento possível do mecanismo de *cache* na rede e a redução da complexidade associada ao controle de acesso. Apesar disso, elas diferem quanto à cardinalidade (na publicação de conteúdos), à intrusividade (isto é, introdução ou modificação de com-

Tabela 1. Propostas para o controle de acesso a conteúdos, organizadas por critérios de cardinalidade, de intrusividade e de proteção ao conteúdo.

Propostas	Cardinalidade		Intrusividade		Proteção do Conteúdo	
	um publicador	vários publicadores	não intrusivas	intrusivas	criptografia simétrica	criptografia assimétrica
Misra et al. [Misra et al. 2013]	x		x		x	
Papanis et al. [Papanis et al. 2014]	x		x		x	
Wood e Uzun [Wood and Uzun 2014]	x			x	x	
Mannes et al. [Mannes et al. 2014]	x		x			x
Singh et al. [Singh et al. 2012]		x		x	nenhum	
Fotiou et al. [Fotiou et al. 2012]		x		x	x	
Hamdane et al. [Hamdane et al. 2013]		x		x	x	
Ghali et al. [Ghali et al. 2015]		x		x	nenhum	

ponentes na rede) e à forma como o conteúdo em si é protegido. A Tabela 1 apresenta uma visão geral das soluções propostas, organizadas segundo esses critérios.

Misra *et al.* [Misra et al. 2013] e Papanis *et al.* [Papanis et al. 2014] propõem que os provedores protejam o conteúdo empregando criptografia simétrica. Enquanto Misra *et al.* empregam o conceito de criptografia em *broadcast* para a distribuição das chaves de acesso ao conteúdo, por sua vez, Papanis *et al.* utilizam o mecanismo CP-ABE (*Ciphertext-Policy Attribute-based Encryption*) [Bethencourt et al. 2007]. Essas soluções tiram bastante proveito do mecanismo de *cache* na rede, por utilizarem criptografia simétrica para proteger o conteúdo. No entanto, elas permitem apenas um publicador no grupo de compartilhamento seguro. Uma vez que cada publicador precisa criar um par de chaves para cada usuário que deve ter acesso aos conteúdos, no cenário em que todos são publicadores, o número de pares de chaves necessárias é proporcional a $\binom{n}{2}$.

Wood e Uzun [Wood and Uzun 2014] e Mannes *et al.* [Mannes et al. 2014] seguem o mesmo modelo baseado em apenas um publicador. No entanto, elas empregam a técnica de reencriptação por *proxy*, proposta orginalmente por Ateniese *et al.* [Ateniese et al. 2006]. Essa técnica utiliza entidades de *proxy* na rede para transformar um conteúdo criptografado, com a chave pública do provedor, em outro conteúdo criptografado, agora com a chave pública do usuário. A principal diferença entre essas soluções reside no critério de intrusividade: enquanto na de Wood e Uzun, depende-se de nodos intermediários para atuar como redistribuidores de chaves de reencriptação, na proposta de Mannes *et al.*, possibilita-se que os próprios publicadores implementem esse papel. Embora menos intrusiva, essa última requer que o publicador do conteúdo esteja permanentemente disponível para criar e distribuir as chaves de reencriptação sempre que o conteúdo for acessado.

As demais soluções propostas avançam no critério de cardinalidade, permitindo vários publicadores no mesmo grupo de compartilhamento seguro, assim, evitando o problema de explosão combinatorial de chaves. Elas dependem, porém, de entidades adicionais para armazenar conteúdos e/ou realizar controle de acesso, o que implica em modificações na arquitetura ICN ou na dependência da disponibilidade dessas entidades. As soluções de Singh *et al.* [Singh et al. 2012] e Ghali *et al.* [Ghali et al. 2015] são particularmente dependentes do comportamento honesto dessas entidades; caso sejam subvertidas, a privacidade dos conteúdos controlados pelas mesmas poderá ser comprometida.

Um aspecto importante que pode ser observado na Tabela 1 é que nenhuma das propostas existentes reúne as características de não intrusividade e de suporte a múltiplos publicadores. Na seção a seguir, apresenta-se um modelo de segurança que satisfaz esses

requisitos, sem dependência de componentes específicos na arquitetura subjacente e sem aumentar a complexidade do processo de gerenciamento de chaves.

3. Habilitando Grupos com Múltiplos Publicadores em ICN

A Figura 1 apresenta uma visão geral do modelo e da arquitetura que o apoia, destacando ainda os atores e componentes envolvidos. O compartilhamento seguro de conteúdos se inicia quando um usuário interage com uma instância da *Aplicação ICN*, executando na sua própria estação local para *criar um grupo*. A aplicação ICN corresponde a um *software* (o equivalente a um navegador *web*) que permite o compartilhamento de conteúdos via paradigma ICN, estendido para suportar o modelo de segurança proposto.

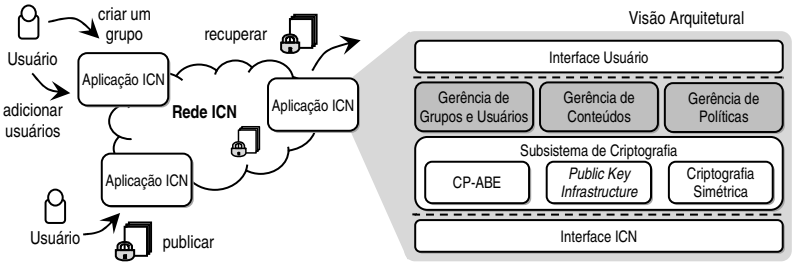


Figura 1. Visão arquitetural da solução proposta.

Apenas o usuário que criou o grupo (denominado de *administrador* no restante do artigo) poderá *adicionar usuários* ao mesmo. Conforme será discutido na próxima subseção, a adição de um usuário consiste na criação de uma credencial de membro (uma chave privada) e na entrega dessa credencial para o usuário. Essa entrega deve ocorrer via compartilhamento seguro, por exemplo, com o uso de criptografia assimétrica. Uma vez habilitado como membro do grupo, o usuário poderá compartilhar conteúdos (isto é, *publicar* e *recuperar* conteúdos) de forma segura com os demais membros.

A visão arquitetural apresentada na parte direita da Figura 1 destaca (em cinza) os componentes que fazem parte da proposta. O componente *Gerência de Grupos e Usuários* reúne as funcionalidades para criação de grupos e adição de membros. O componente *Gerência de Conteúdos* está relacionado com a publicação e recuperação segura de conteúdos. Por fim, o componente *Gerência de Políticas* possibilita o controle fino de acesso aos conteúdos, apoiando, por exemplo, a concessão e a revogação de acesso. Esses componentes são apoiados por um subsistema de criptografia, composto por um mecanismo de criptografia simétrica, uma solução de infraestrutura de chaves públicas (*Public Key Infrastructure, PKI*) e um mecanismo de criptografia baseada em atributos (CP-ABE) [Bethencourt et al. 2007] (doravante referido como *componente CP-ABE*). Observe que os componentes da solução proposta acomodam-se exclusivamente entre as camadas de interface com o usuário e com a rede ICN, sendo restritos portanto ao *software* que executa na estação do usuário.

As Subseções 3.1, 3.2 e 3.3, a seguir, descrevem em detalhes as funcionalidades proporcionadas por cada um dos componentes destacados na visão arquitetural da Figura 1. A Subseção 3.4 encerra a apresentação da proposta, discutindo possíveis estratégias de ataque contra a solução. Para a explicação que segue adota-se um conjunto de notações e convenções sumarizado na Tabela 2.

3.1. Gerência de Grupos e Usuários

A Figura 2 ilustra a dinâmica do processo de manutenção de grupos de compartilhamento seguro, destacando as atividades de criação de grupos e de adição de usuários ao mesmo.

Tabela 2. Glossário de notações relacionadas ao modelo de segurança proposto.

Notação	Definição formal	Descrição
Entidades e conjuntos		
C		Conteúdo original
G		Grupo de compartilhamento seguro
U		Usuário (membro do grupo)
$\mathbb{P}$		Política de acesso
$\mathcal{L}_G$		Conjunto de atributos existentes no grupo G
$\mathcal{L}_{U,G} \subseteq \mathcal{L}_G$		Conjunto de atributos do usuário U no grupo G
Chaves criptográficas		
K_s		Chave de criptografia simétrica
K_U		Chave pública do usuário U
K_U^{-1}		Chave privada do usuário U
K_G		Chave pública do grupo de compartilhamento seguro G
M_G		Chave mestra do grupo de compartilhamento seguro G
$K_{\mathcal{L}_{U,G}}^{-1}$		Chave privada do usuário U no grupo G
Funções criptográficas		
$\{X\}_{K_x}$		Elemento X cifrado usando a chave K_x (simétrica ou assimétrica)
$\{X\}_{(K_G,P)}$		Elemento X cifrado usando a chave do grupo K_G e a política P
Modelo de segurança		
$\widehat{X}$		Identificador do elemento X
C_P	$C_P = \langle \{C\}_{K_s}, \widehat{H_C} \rangle$	Conteúdo C protegido
H_C	$H_C = \langle \{K_s\}_{(K_G,\mathbb{P})}, \widehat{K_G} \rangle$	Bloco habilitador de um conteúdo protegido C_P

Criação de grupo. Como brevemente mencionado anteriormente, o administrador inicia esse processo ao interagir com a aplicação ICN (fluxo 1 na Figura 2). Esse processo compreende basicamente a criação de um par de chaves pública K_G e mestra M_G para o grupo, o que é feito com o apoio do componente CP-ABE. O grupo passa a existir na rede a partir do momento que se dissemina a chave pública K_G na rede, na forma de um objeto, utilizando como identificador o nome do grupo (fluxo 2). A chave mestra M_G é mantida em segredo pelo administrador.

Cada grupo possui um conjunto de atributos $\mathcal{L}_G$, que são utilizados para descrever os usuários membros. Os atributos são cadeias de caracteres de tamanho livre, definidos pelo administrador e existentes somente no escopo daquele grupo. Por exemplo, suponha um grupo composto por integrantes de uma comunidade acadêmica. Alguns atributos possíveis são "professor", "graduando", "mestrando", "doutorando" e "pós-doutorando". Note que não há uma regra geral para a formação dos atributos. Da mesma forma, a semântica dos atributos é dada pelo contexto do grupo. A lista de atributos também deve ser publicada como um objeto na rede (fluxo 3).

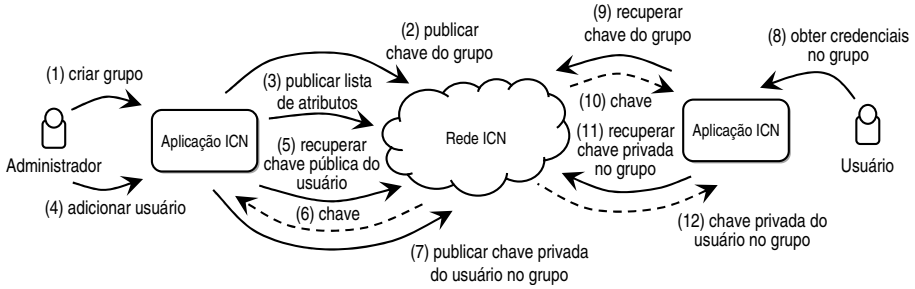


Figura 2. Manutenção de grupos de compartilhamento seguro.

Adição de usuário a um grupo. O administrador inicia esse processo, por intermédio da aplicação ICN (fluxo 4), ao informar os atributos que o novo membro possuirá. Esse processo se desdobra em três passos: (*i*) criar a chave privada do usuário no

grupo $K^{-1}_{\mathcal{L}_{U,G}}$; (*ii*) publicar a chave $K^{-1}_{\mathcal{L}_{U,G}}$ de forma segura na rede, de modo que apenas o usuário adicionado possa recuperá-la; e (*iii*) recuperar a chave $K^{-1}_{\mathcal{L}_{U,G}}$ da rede (esse último passo feito pelo usuário adicionado). Esses passos são descritos em detalhes a seguir.

A chave privada $K^{-1}_{\mathcal{L}_{U,G}}$ é criada com o apoio do componente CP-ABE. Para tal, o administrador deve especificar um conjunto de atributos $\mathcal{L}_{U,G} \subseteq \mathcal{L}_G$ para o usuário. A criação de $K^{-1}_{\mathcal{L}_{U,G}}$ requer também a chave mestra do grupo M_G. Após criada, disponibiliza-se a chave $K^{-1}_{\mathcal{L}_{U,G}}$ (com os atributos $\mathcal{L}_{U,G}$ incorporados) na rede, para que o usuário alvo possa recuperá-la. A entrega deve ocorrer de forma privativa, visto que a posse de $K^{-1}_{\mathcal{L}_{U,G}}$ materializa a participação no grupo. Em outras palavras, $K^{-1}_{\mathcal{L}_{U,G}}$ será empregada para recuperar conteúdos protegidos no grupo (conforme discutido na próxima subseção). Para realizar essa entrega, o administrador precisa recuperar da rede a chave pública K_U do usuário e verificá-la via mecanismo de infraestrutura de chave pública (fluxos 5 e 6). A chave $K^{-1}_{\mathcal{L}_{U,G}}$ é criptografada usando K_U, assim, gerando a chave criptografada $\{K^{-1}_{\mathcal{L}_{U,G}}\}_{K_U}$, a qual é publicada como um objeto na rede (fluxo 7). Por fim, o usuário alvo precisa recuperar as chaves $K^{-1}_{\mathcal{L}_{U,G}}$ e K_G da rede, para poder utilizá-las na publicação e recuperação de conteúdos no grupo. O usuário inicia esse procedimento (fluxo 8) especificando o nome do grupo para recuperar esses objetos. A aplicação recupera, então, a chave pública do grupo K_G (fluxos 9 e 10) e a chave privada do usuário no grupo, criptografada $\{K^{-1}_{\mathcal{L}_{U,G}}\}_{K_U}$ (fluxos 11 e 12). A chave privada no grupo é descriptografada usando a própria chave privada do usuário K^{-1}_U. A partir de então, ele está habilitado para publicar e recuperar conteúdos no grupo. Note que o administrador, caso deseje publicar e recuperar conteúdos do grupo, também precisa criar a sua própria chave $K^{-1}_{\mathcal{L}_{U,G}}$.

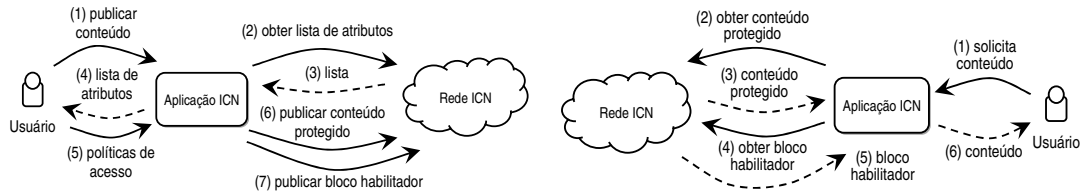
3.2. Gerência de Conteúdos

A gerência de conteúdos reúne todos os procedimentos necessários para a publicação e a recuperação segura de conteúdos na rede. Esses procedimentos se apoiam em dois elementos importantes, o conteúdo protegido e o bloco habilitador.

Conteúdo protegido e bloco habilitador. O primeiro corresponde a um conteúdo cifrado usando criptografia simétrica, enquanto que o segundo contém a chave necessária para decifrar um dado conteúdo. No modelo proposto, um conteúdo protegido possui um (e apenas um) bloco habilitador correspondente. Formalmente, um conteúdo protegido é uma tupla $C_P = \langle \{C\}_{K_s}, \widehat{H_C} \rangle$, onde $\{C\}_{K_s}$ corresponde ao conteúdo original C, cifrado usando uma chave simétrica K_s, e $\widehat{H_C}$ é o identificador do bloco habilitador desse conteúdo protegido. Um bloco habilitador é uma tupla $H_C = \langle \{K_s\}_{(K_G,\mathbb{P})}, \widehat{K_G} \rangle$, onde $\{K_s\}_{(K_G,\mathbb{P})}$ corresponde à chave K_s (usada para cifrar C) cifrada usando (*i*) a chave pública K_G do grupo e (*ii*) uma política de acesso $\mathbb{P}$, e $\widehat{K_G}$ é o identificador da chave pública do grupo. Observe que esse projeto possibilita que vários conteúdos sejam protegidos por um mesmo bloco habilitador. Essa característica pode ser conveniente quando se desejar publicar múltiplos conteúdos usando uma política de acesso única.

O processo para construir um bloco habilitador compreende (*i*) a definição da chave simétrica que será usada para cifrar o conteúdo e (*ii*) a especificação (pelo usuário) da *política de controle de acesso* $\mathbb{P}$. Em relação à chave simétrica, ela pode ser gerada automaticamente, pela aplicação ICN, ou informada pelo usuário. Sobre a política $\mathbb{P}$, ela determinará que usuários do grupo estarão autorizados a decifrar o conteúdo e é feita envolvendo elementos da lista de atributos do grupo $\mathcal{L}_G$.

Para ilustrar o conceito de políticas de acesso, suponha o grupo de compartilha-



(a) Publicação (b) Recuperação

Figura 3. Sequência de passos para publicação e recuperação de conteúdos.

mento de conteúdos acadêmicos nos quais os usuários possuem um (ou mais) dos seguintes atributos: $\mathcal{L}_G = \{\text{professor, graduando, mestrando, doutorando, pos-doutorando}\}$. O usuário pode, por exemplo, especificar uma política $\mathbb{P} = \{\text{professor } \textbf{ou} \text{ graduando}\}$. Nesse caso, a descriptografia baseada em atributos (usando o componente CP-ABE) poderá ser realizada apenas pelos usuários detentores do atributo "professor" ou "graduando" (ou ambos). A metodologia para a formação dessas políticas será discutida mais detalhadamente na Subseção 3.3. Uma vez determinada $\mathbb{P}$, emprega-se o componente CP-ABE para cifrar K_s. Essa cifragem é executada usando a chave pública K_G do grupo e a política $\mathbb{P}$. A chave cifrada $\{K_s\}_{(K_G,\mathbb{P})}$ é, então, encapsulada em H_C.

Após construído o bloco habilitador, o conteúdo protegido pode então ser formado. A sua construção compreende a cifragem do conteúdo original C a ser disseminado na rede, para isso, utilizando-se a chave simétrica K_s encapsulada no bloco habilitador que será associado a esse conteúdo.

Publicação de conteúdos. A Figura 3(a) ilustra a dinâmica do processo de publicação de um conteúdo no grupo. O usuário inicia esse processo ao interagir com a aplicação ICN (fluxo 1 na Figura 3(a)), informando o conteúdo C a ser publicado. Nesse momento, cinco passos são executados. Primeiro, a aplicação recorre à rede para obter a lista atualizada dos atributos do grupo $\mathcal{L}_G$ (fluxos 2 e 3) e disponibiliza-os ao usuário. Em seguida, o usuário elabora a política de acesso $\mathbb{P}$, conforme as restrições de acesso desejadas (fluxos 4 e 5). O terceiro passo, realizado pela aplicação, consiste em criptografar o conteúdo C usando uma chave simétrica K_s. O quarto passo corresponde à construção do bloco habilitador H_C do conteúdo, conforme discutido anteriormente. No último passo, o conteúdo protegido C_P é construído, encapsulando o conteúdo cifrado $\{C\}_{K_s}$ e o identificador para o bloco habilitador $\widehat{H_C}$. Por fim, ambos, conteúdo protegido C_P e bloco habilitador H_C, são publicados na rede (fluxos 6 e 7).

Recuperação de conteúdos. O processo de recuperação, ilustrado na Figura 3(b), inicia-se quando o usuário solicita um conteúdo (fluxo 1). A aplicação solicita à rede o conteúdo protegido C_P correspondente (fluxos 2 e 3), o qual obtém-se da fonte mais próxima. Ao abrir C_P, a aplicação identifica qual bloco habilitador H_C está relacionado a esse conteúdo (por meio do identificador $\widehat{H_C}$ presente na tupla). A aplicação solicita então H_C à rede (fluxos 4 e 5) para recuperar a chave simétrica criptografada $\{K_s\}_{(K_G,\mathbb{P})}$. A chave simétrica $\{K_s\}_{(K_G,\mathbb{P})}$ recuperada é submetida ao componente CP-ABE para decifragem. Para isso, o usuário utiliza a sua chave privada no grupo $K^{-1}_{\mathcal{L}_{U,G}}$. A chave $\{K_s\}_{(K_G,\mathbb{P})}$ é descriptografada *se e somente se* a política de acesso $\mathbb{P}$ usada para cifrá-la for compatível com os atributos utilizados pelo administrador na criação de $K^{-1}_{\mathcal{L}_{U,G}}$ (ao adicionar o usuário no grupo). Caso seja descriptografado com sucesso, o conteúdo é entregue ao usuário (fluxo 6).

Tabela 3. Regras para a definição de políticas de acesso a conteúdos.

⟨política⟩ ::= ⟨atributo⟩ | '(' ⟨política⟩ ')'
| ⟨atributo⟩ e ⟨política⟩ | ⟨atributo⟩ ou ⟨política⟩
| ⟨atributo⟩ = ⟨inteiro⟩ | ⟨atributo⟩ < ⟨inteiro⟩ | ⟨atributo⟩ > ⟨inteiro⟩
| ⟨inteiro⟩ de '(' ⟨coleção⟩ ')'

⟨coleção⟩ ::= ⟨política⟩ ',' ⟨política⟩ | ⟨coleção⟩

3.3. Gerência de Políticas

O modelo de segurança, por meio do componente *Gerência de Políticas*, permite determinar quando e quais usuários podem ter acesso aos conteúdos publicados. Em outras palavras, o modelo reúne mecanismos que permitem conceder, limitar e revogar autorizações de acesso a conteúdos, com base em políticas.

A composição de uma política envolve operadores relacionais ($>$, $<$ e $=$) e lógicos (**e** e **ou**). Com o apoio do componente CP-ABE [Bethencourt et al. 2007], esses operadores permitem determinar quando e quais usuários têm acesso ao conteúdo. A Tabela 3 apresenta o conjunto de regras que regem o processo de construção de políticas de acesso a conteúdo. Nesse conjunto, $<$*atributo*$> \in \mathcal{L}_G$ e $<$*inteiro*$> \in \mathbb{N}$.

Concessão de acesso. Ela consiste basicamente em definir uma política que um determinado conjunto de usuários do grupo deve satisfazer para decifrar um conteúdo. Observe que uma política pode ser formada por apenas um atributo. Nesse caso, para ter acesso ao conteúdo, a chave privada do usuário no grupo $K^{-1}_{\mathcal{L}_{U,G}}$ deve atender à restrição descrita pela política. Por exemplo, suponha dois usuários: *joão* com os atributos $\mathcal{L}_{\text{joão},G} = \{\text{professor, pesquisador}\}$, e *josé* com os atributos $\mathcal{L}_{\text{josé},G} = \{\text{aluno, pesquisador}\}$. Uma política de acesso $\mathbb{P}_1 = \{\text{professor}\}$ permite acesso ao conteúdo apenas para *joão*. A política de acesso $\mathbb{P}_2 = \{\text{pesquisador}\}$, por sua vez, permite o acesso a ambos. As regras não permitem a formação de políticas "coringa", isto é, para acesso universal. Uma forma de alcançar todos os usuários é citar um a um os atributos dos mesmos na política de acesso. Alternativamente, o administrador pode definir um atributo comum aos usuários (ex. "todos"); assim, cada publicador poderá usá-lo em políticas que visem ao acesso universal.

Os atributos (dos usuários e de políticas) podem ainda ser valorados. Eles podem ser criados para indicar, por exemplo, o nível do usuário na hierarquia de uma corporação. Suponha o usuário *joão* com o atributo "nível = 5" e *josé* com o atributo "nível = 2". Caso deseje-se publicar um conteúdo somente para os usuários com nível 3 ou superior (no caso, *joão*), basta definir a política $\mathbb{P} = \{\text{nível} > 2\}$ (assumindo que os níveis de hierarquia são dados por números discretos).

Revogação de acesso. A revogação baseia-se na possibilidade de publicar uma versão mais recente de um bloco habilitador (por exemplo, quando a versão existente expirar na *cache* dos roteadores). Assim, o usuário pode reformular a política daquele bloco para restringir o acesso de algum usuário particular. Há duas estratégias que podem ser empregadas. A primeira é definir um atributo único para cada usuário. Nesse caso, a revogação compreenderia selecionar todos os usuários exceto aquele(s) cujo acesso deve ser revogado. Supondo os usuários *joão*, *josé*, *maria* e *fátima*, e um conteúdo publicado com $\mathbb{P} = \{\text{todos}\}$, revogar o acesso de *joão* a esse conteúdo requer que a política seja reformulada para $\mathbb{P}' = \{1 \,\text{de}\, (\text{jose, maria, fatima})\}$. A complexidade de se definir essa restrição para grupos com dezenas de usuários ou mais pode ser trivialmente resolvida na

interface com o usuário, não sendo necessário transportá-la para o modelo.

A segunda forma de implementar revogação é por meio de expiração, usando o mecanismo de comparação de valores de atributos. Para ilustrar, suponha que *joão* possui o atributo "criado = 1435708800" (2015-07-01 00:00:00) e *maria*, "criado = 1446336000" (2015-11-01 00:00:00). A semântica desses atributos corresponde à data e hora (em *timestamp*) que cada um foi adicionado ao grupo. Agora, suponha um conteúdo com a política $\mathbb{P} = \{\text{criado} > 1420070400\}$. Ela garante acesso apenas aos usuários adicionados ao grupo após 1º de janeiro, o que se aplica a *joão* e *maria*. O acesso de *joão* pode ser revogado por expiração, nesse caso, ao se publicar um novo bloco habilitador com $\mathbb{P}' = \{\text{criado} > 1443657600\}$ (1º de outubro). Destaca-se que ambas estratégias podem ser usadas em conjunto, permitindo revogação a curto e longo prazo.

3.4. Possíveis Estratégias de Ataque

O modelo apresentado para compartilhamento seguro de conteúdos foi projetado considerando três premissas básicas: (*i*) o administrador do grupo é confiável; (*ii*) os membros do grupo mantêm em segredo suas respectivas chaves privadas no grupo; e (*iii*) o membro com acesso a um conteúdo protegido não divulga a chave simétrica usada para cifrá-lo.

As implicações das premissas enumeradas acima são descritas a seguir. A primeira estabelece que a concessão de atributos aos usuários membros do grupo é feita de forma confiável. Ou seja, o administrador não deturpará o uso dos atributos ao atribuí-los aos membros, por exemplo, conferindo atributo(s) a um adversário ou mesmo a usuários que não sejam compatíveis com aquele(s) atributo(s). Essa premissa é similar à confiabilidade depositada no gerente de projetos estratégicos e/ou sensíveis, por exemplo, de um *software open-source* (no qual a admissão de um adversário ao time de desenvolvedores pode levar à inclusão de código malicioso no *software* desenvolvido). A segunda premissa está relacionada à segurança dos conteúdos que são acessíveis por determinados usuários. Essa premissa é equivalente à guarda das credenciais de acesso que um usuário possui em um sistema (por exemplo, a chave para um servidor *ssh* ou a senha para um portal de conteúdos pagos). Finalmente, a terceira premissa implica na segurança dos conteúdos protegidos que foram acessados. Nesse caso, o vazamento da chave simétrica seria equivalente ao ato de vazar o próprio conteúdo.

Considerando essas premissas, um atacante pode burlar o modelo de segurança proposto apenas se comprometer (via acesso físico ou remoto) a própria estação do administrador/usuário para subtrair a chave privada do grupo ou as chaves privadas dos usuários. Observe que esse ataque foge ao escopo do artigo, uma vez que a proteção contra o mesmo requer mecanismos que garantam a segurança da própria estação do administrador/usuário. Assumindo a segurança das mesmas, o modelo mantém-se resiliente mesmo que o atacante possua acesso privilegiado a quaisquer elementos da rede ou nela disponíveis (roteadores, blocos habilitadores, chaves públicas, listas de atributos, etc.).

O modelo proposto é robusto a ataques em conluio. Por exemplo, suponha um usuário com o atributo professor e outro com o atributo aluno. Mesmo que esses usuários atuem em conjunto, eles não podem decifrar um conteúdo protegido com a política $\mathbb{P} = \{\text{professor } \mathbf{e} \text{ aluno}\}$, visto que a política requer que o mesmo membro possua ambos os atributos, simultaneamente. Conforme discutido anteriormente, apenas usuários pertencentes ao grupo e que satisfaçam integralmente as políticas de acesso definidas, podem acessar conteúdos publicados. Por fim, o modelo não impede que usuários não pertencentes ao grupo publiquem conteúdos protegidos no mesmo. Isso é possível visto que a publicação requer apenas a chave pública e a lista de atributos do grupo, ambos disponíveis em claro na rede. Os membros do grupo podem evitar o acesso a conteúdos

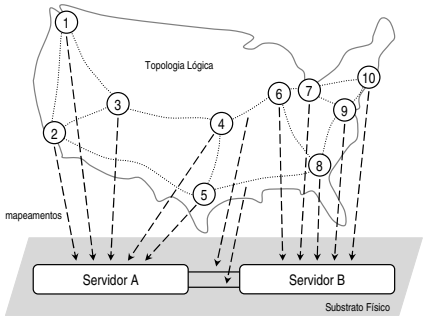


Figura 4. Topologia base da avaliação.

Tabela 4. Cenários avaliados.

Parâmetros	Cenários avaliados	
	A	B
Quantidade de usuários	10	30
Tamanho do arquivo	100MB	100MB
Arquivos publicados	10	30
Capacidade da *cache*	1GB	1GB
Expiração da *cache*	1 hora	1 hora
Tamanho do *chunk*	4KB	4KB
Popularidade dos conteúdos	Zipf [1] ($s = 2.0$)	Zipf [1] ($s = 2.0$)

[1] Segundo *Pentikousis et al.* [Pentikousis et al. 2015]

indesejados verificando a origem dos mesmos usando, por exemplo, mecanismos de auto-certificação de conteúdos da própria rede [Jacobson et al. 2012].

4. Avaliação

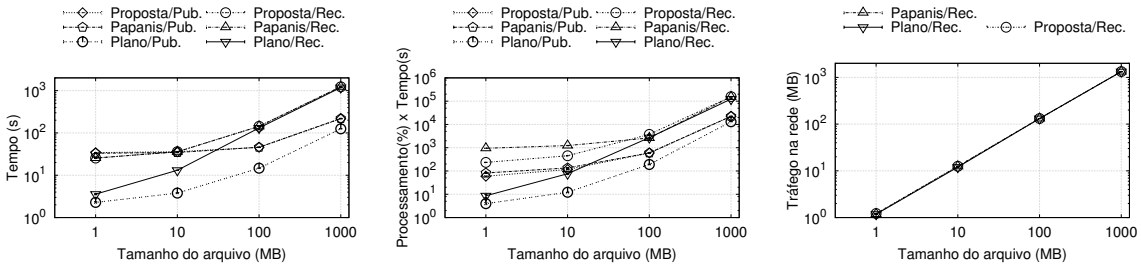
Para aferir a eficácia e eficiência do modelo proposto, uma série de experimentos foi realizada em um ambiente controlado. Os experimentos visaram verificar a escalabilidade do modelo, o custo de operação e o impacto à qualidade de experiência (QoE) dos usuários, em cenários com um número variado de usuários atuando como publicadores e recuperadores. Para comparação, considerou-se a solução de Papanis *et al.* [Papanis et al. 2014] e uma solução de compartilhamento seguro de conteúdos baseada no algoritmo RSA.

4.1. Configuração do Ambiente e Cenários de Avaliação

O modelo proposto foi implementado sobre a arquitetura CCN (*Content Centric Networking*) [Jacobson et al. 2012], usando como base o *software* CCNx 0.8.2 executando sobre a máquina virtual Java SE versão 8. Para o mecanismo de criptografia baseada em atributos, foi utilizado o *software cpabe* 0.11 [Bethencourt et al. 2015]. Visando seguir o padrão para gerenciamento de chaves proposto para arquiteturas ICN [Bian et al. 2013], cada conteúdo protegido é acompanhado de um respectivo metadado, que contém o identificador do bloco habilitador correspondente e a validade do conteúdo, entre outros. Da mesma forma, cada bloco habilitador é acompanhado por um respectivo metadado.

O substrato físico usado nos experimentos compreendeu dois servidores, cada um equipado com 1 processador Intel Xeon E5-2420 (1.9GHz, 12 Threads e 15MB *cache*), 32GB de memória RAM (1333MHz), 1 HD SAS (1TB de capacidade) e 2 interfaces de rede Gigabit Ethernet. Ambos possuem Debian/Linux 7.7 (kernel 3.14.21) e Hipervisor Xen instalados. Os servidores foram conectados diretamente entre si usando dois cabos Ethernet. A topologia lógica usada para a avaliação, um subconjunto da Internet2, é ilustrada na Figura 4. Os mapeamentos dos elementos lógicos para o substrato físico também são apresentados na figura. Cada nodo lógico na topologia corresponde a uma máquina virtual; cada máquina foi instanciada com as seguintes configurações: 2 processadores virtuais, 2 GB de RAM e 40 GB de disco. Os enlaces entre os nodos foram emulados empregando o *software bridge-utils* versão 1.5, todos com velocidade de $\approx$ 98Mbps.

Para a avaliação experimental foram considerados dois cenários, cujos parâmetros mais relevantes são sumarizados na Tabela 4. Por simplificação, todos os conteúdos foram publicados usando uma política de acesso universal (ou seja, qualquer usuário membro do grupo pode decifrá-lo). Essa decisão foi embasada em experimentos preliminares, os quais permitiram observar que a quantidade de atributos não tem efeito relevante sobre custos (tempo de publicação/recuperação, tráfego de rede, etc.) do modelo proposto. Por



(a) Tempo (b) Carga de processamento (c) Tráfego na rede

Figura 5. Custos da solução proposta considerando tempos de publicação e de recuperação, carga de processamento e tráfego gerado na rede.

fim, para cada experimento, foram realizadas 30 execuções e calculado o intervalo de confiança com nível de significância $\alpha = 0.05$.

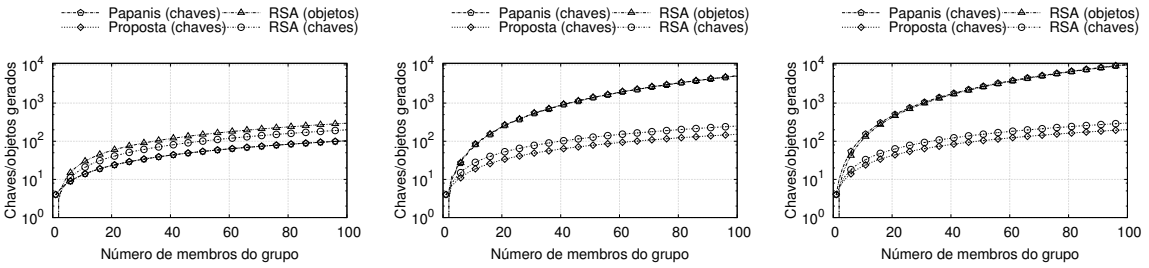
4.2. Custos da Solução Proposta

A primeira parte da avaliação compreende uma análise dos custos da solução em um ambiente isolado, formado por apenas um publicador e um recuperador. Para isso, utilizou-se subconjunto da topologia ilustrada na Figura 4, formada pelos nodos 4 (publicador) e 6 (recuperador). A Figura 5 apresenta uma visão geral dos resultados obtidos para a publicação (curvas "Pub.") e a recuperação (curvas "Rec.") de conteúdos. Para fins de comparação, considerou-se a solução de Papanis *et al.* (curvas "Papanis") e uma solução sem mecanismos de segurança (curvas "Plano"). Por legibilidade, os gráficos são apresentados com o eixo y em escala logarítmica.

A principal conclusão que se pode tirar a partir dos resultados da Figura 5 é a de que o sobrecusto da solução proposta é marginal quando comparado a Papanis *et al.* Focando no tempo médio de disseminação de conteúdos (Figura 5(a)), por exemplo, a solução proposta foi inclusive 0,6% mais eficiente em média na publicação. Em relação aos custos de processamento (Figura 5(b)), esses são ligeiramente maiores na solução proposta (0,5% na publicação e 1,4% na recuperação). Por fim, observa-se que a medição do tráfego gerado na rede (Figura 5(c)) indica desempenhos similares de ambas as soluções.

Quando comparados à solução sem mecanismos de segurança, note que os custos são amortizados de forma proporcional ao tamanho do conteúdo publicado. Esses resultados sugerem que a solução proposta incorre em impacto relativamente pequeno para a qualidade de experiência (QoE) dos usuários. Na publicação, por exemplo, o sobrecusto de tempo diminui de 1.400% (diferença do custo entre a solução proposta e a solução "Plano"), em média (com conteúdos de 1MB) para 72% (1000MB). Nessa comparação, o sobrecusto médio foi de apenas 8% no tempo de recuperação de conteúdos (aspecto de maior importância para a QoE de grande parte dos usuários). Os resultados obtidos para o modelo proposto são inclusive similares ao observado para Papanis *et al.*

É importante mencionar que os custos adicionais de processamento e de tempo devem-se ao uso de criptografia para cifrar/decifrar o conteúdo e as chaves de acesso. Quando não há solução de segurança sendo utilizada, o processamento e o tempo referem-se apenas à publicação/recuperação do conteúdo na rede. Sobre o tráfego gerado na rede (Figura 5(c)), o sobrecusto foi constante e marginal, correspondendo principalmente ao bloco habilitador do conteúdo protegido (que também é disseminado na rede).



(a) 1 publicador (b) $n/2$ publicadores (c) n publicadores

Figura 6. Número de chaves/objetos necessários para a troca de conteúdos.

4.3. Quantidade de Chaves e de Objetos Gerados

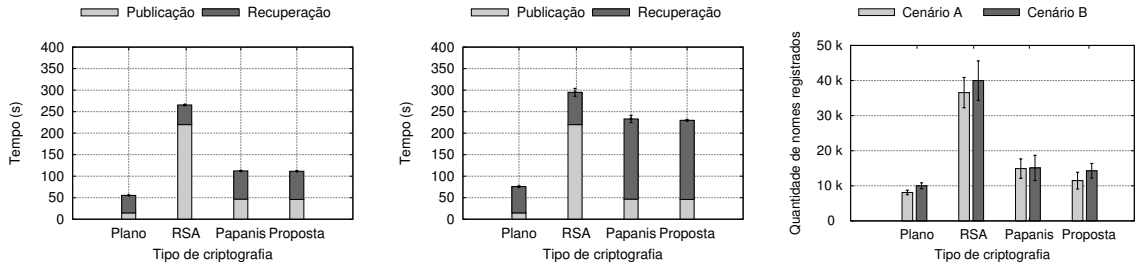
Outro aspecto observado está relacionado à quantidade de chaves/objetos necessários para a disseminação segura de conteúdos, nas situações em que um, metade e todos os usuários atuam como publicadores na rede, respectivamente. Nessa análise, realizada por meio de uma formulação matemática, foram consideradas além da solução proposta: a de Papanis *et al.* [Papanis et al. 2014], e a baseada no RSA. No caso de Papanis *et al.*, ela é instanciada para cada usuário publicador. No modelo baseado no RSA, (*i*) cada usuário possui um par de chaves pública e privada; (*ii*) cada conteúdo é cifrado usando uma chave simétrica única; e (*iii*) a chave do conteúdo é cifrada usando a chave pública de cada usuário alvo.

Os resultados dessa avaliação são apresentados na Figura 6 (o eixo y é apresentado em escala logarítmica). Observa-se que a solução proposta requer o menor número de chaves criptográficas, em comparação com Papanis *et al.* e RSA. Além disso, a solução proposta mantém a proporcionalidade do número das chaves relativas ao número de usuários pertencentes ao grupo, independentemente do número de publicadores. Mais importante, o número de chaves necessárias/objetos publicados aumenta significativamente para as duas últimas. Para Papanis *et al.*, observa-se um aumento de até 9.900%, em contraste com 96% na solução proposta. Em relação a Papanis *et al.*, esse aumento se relaciona ao problema da explosão combinatória de chaves. Embora no cenário usando RSA o número de chaves permaneça relativamente constante, o número de objetos publicados na rede cresce significativamente. O motivo é que, embora a chave simétrica seja única para cada conteúdo, ela precisa ser criptografada individualmente para cada usuário alvo, de modo a garantir que apenas usuários autorizados possam acessar o conteúdo.

4.4. Tempos de Disseminação e Quantidade de Objetos Registrados

O objetivo dessa avaliação, cujos resultados são sumarizados na Figura 7, foi aferir o desempenho da solução proposta, mais especificamente o tempo para disseminação de conteúdos e a sobrecarga à *Forward Information Base* (FIB) dos roteadores, em um ambiente com múltiplos publicadores e consumidores. Essa avaliação usou como base a topologia completa ilustrada na Figura 4 e os cenários sumarizados na Tabela 4. Cada usuário publica 1 e recupera n conteúdos, ou seja, são publicados 10 conteúdos no cenário A e 30 no cenário B. Para comparação, foram empregadas a solução de Papanis *et al.*, uma baseada no algoritmo RSA e outra sem mecanismos de segurança ("Plano").

Observe, nas Figuras 7(a) e 7(b), que a qualidade de experiência (QoE) do usuário (medida pelo tempo necessário para disseminação do conteúdo) é marginalmente afetada na nossa solução, comparado à Papanis *et al.* Mais importante, ela é substancialmente melhor quando comparada à solução baseada no RSA. Esse desempenho é alcançado causando relativamente menos impacto à rede, conforme pode ser observado na Figura 7(c).



(a) Tempos medidos no cenário A (b) Tempos medidos no cenário B (c) FIB nos cenários A e B

Figura 7. Tempo de publicação/recuperação dos conteúdos e quantidade de nomes registrados na FIB, para os cenários A e B.

O tempo relativamente maior de recuperação nas soluções proposta e a de Papanis *et al.* explica-se pelo fato de que a solução baseada no RSA não requer metadados de conteúdo ou de bloco habilitador. Em outras palavras, cada usuário pode localizar diretamente os conteúdos e suas respectivas chaves sem a necessidade de obter os metadados dos mesmos, sendo, portanto, irrelevante publicá-los. Por outro lado, o tempo de publicação é significativamente maior no RSA, visto que n versões criptografadas da chave de um mesmo conteúdo devem ser publicadas na rede, uma para cada usuário alvo. Essa característica se reflete no gráfico da Figura 7(c), com a solução proposta reduzindo em até 68% (no cenário B) a quantidade de nomes registrados na FIB dos roteadores.

5. Considerações Finais

A publicação segura de conteúdos em ICN é uma realidade, com diversas soluções que oferecem os mais variados níveis de controle de acesso. Apesar de promissoras, algumas soluções causam uma sobrecarga significativa na rede ao tornar o processo de gerência (e de distribuição) de chaves combinatorialmente complexo. As soluções que não estão suscetíveis a esse problema, no entanto, são dependentes de arquitetura específica ICN, inserem (ou modificam) componentes na rede e são pouco flexíveis para adoção gradual.

Para suprir essa lacuna, foi apresentada uma nova solução, centrada nos conceitos de grupos de usuários, para o compartilhamento seguro de conteúdos. A partir dos resultados alcançados, foi possível aferir a eficácia e eficiência da solução proposta. Em resumo, esta requer um número comparativamente menor de chaves e objetos na rede (em alguns casos até 97% menos chaves). Esse ganho é alcançado sem degradar a qualidade de experiência do usuário (por exemplo, o tempo necessário para publicar/recuperar conteúdos), ao contrário do que ocorre em outras soluções. Além desses benefícios, a solução proposta pode ser adotada de forma independente e autônoma por um subconjunto de usuários, sem depender de modificações na rede. Por fim, ela permite a publicação e recuperação de conteúdos mesmo que o administrador do grupo (ou o publicador do conteúdo, no caso de recuperação) torne-se indisponível.

Como perspectivas de trabalhos futuros, pretende-se investigar mecanismos para acelerar a disseminação de novas políticas de acesso ao conteúdo na rede, bem como mecanismos para tornar mais simples e eficiente a revogação de acesso a conteúdos.

Referências

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. (2012). A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36.

Ateniese, G., Fu, K., Green, M., and Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1):1–30.

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy (SP 2007)*, pages 321–334.

Bethencourt, J., Sahai, A., and Waters, B. (2015). Advanced crypto software collection. Disponível em: <http://acsc.cs.utexas.edu/cpabe/>. Acesso em: Abril de 2015.

Bian, C., Zhu, Z., Afanasyev, A., Uzun, E., and Zhang, L. (2013). Deploying key management on ndn testbed. Technical report. Disponível em: <http://www.named-data.net/techreport/TR009-publishkey-rev2.pdf>. Acesso em: Junho, 2015.

de Brito, G. M., Velloso, P. B., and Moraes, I. M. (2012). Redes orientadas a conteúdo: Um novo paradigma para a internet. In *Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2012)*, pages 211–264.

Fotiou, N., Marias, G. F., and Polyzos, G. C. (2012). Access control enforcement delegation for information-centric networking architectures. In *ACM SIGCOMM workshop on Information-centric networking (ICN '12)*, pages 85–90.

Ghali, C., Schlosberg, M. A., Tsudik, G., and Wood, C. A. (2015). Interest-based access control for content centric networks (extended version). *CoRR*, abs/1505.06258.

Hamdane, B., Msahli, M., Serhrouchni, A., and El Fatmi, S. (2013). Data-based access control in named data networking. In *Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom 2013)*, pages 531–536.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M., Briggs, N., and Braynard, R. (2012). Networking named content. *Commun. ACM*, 55(1):117–124.

Mannes, E., Maziero, C., Lassance, L. C., and Borges, F. (2014). Controle de acesso baseado em reencriptação por proxy em redes centradas em informação. In *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2014)*.

Misra, S., Tourani, R., and Majd, N. E. (2013). Secure content delivery in information-centric networks: design, implementation, and analyses. In *ACM SIGCOMM workshop on Information-centric networking (ICN '13)*, pages 73–78.

Papanis, J. P., Papapanagiotou, S. I., Mousas, A. S., Lioudakis, G. V., Kaklamani, D. I., and Venieris, I. S. (2014). On the use of attribute-based encryption for multimedia content protection over information-centric networks. *Transactions on Emerging Telecommunications Technologies*, 25(4):422–435.

Pentikousis, K., Ohlman, B., Davies, E., Spirou, S., Boggia, G., and Mahadevan, P. (2015). Information-centric networking: Evaluation methodology draft-irtf-icnrg-evaluation-methodology-03. URL: https://tools.ietf.org/html/draft-irtf-icnrg-evaluation-methodology-03. Acesso em: Outubro de 2015.

Singh, S., Puri, A., Singh, S. S., Vaish, A., and S.Venkatesan (2012). A trust based approach for secure access control in information centric network. *Journal of Information and Network Security*, 1(2):97–104.

Wood, C. and Uzun, E. (2014). Flexible end-to-end content security in ccn. In *11th Consumer Communications and Networking Conference (CCNC 2014)*.

Xylomenos, G., Ververidis, C., Siris, V., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K., and Polyzos, G. (2014). A survey of information-centric networking research. *IEEE Communications Surveys Tutorials*, 16(2):1024–1049.

Q-caching: an integrated reinforcement-learning approach for caching and routing in information-centric networks

Wouter Caarls*, Eduardo Hargreaves, Daniel S. Menasché

[1] Universidade Federal do Rio de Janeiro (UFRJ)
Av. Athos da Silveira Ramos, 149 - Blc C - 21.941-909
Rio de Janeiro, Brasil

wouter@caarls.org,eduardo.hargreaves@ppgi.ufrj.br,sadoc@dcc.ufrj.br

Abstract. *Content delivery, such as video streaming, is one of the most prevalent Internet applications. Although very popular, the continuous growth of such applications poses novel performance and scalability challenges. Information-centric networks put content at the center, and propose novel solutions to such challenges but also pose new questions on the interface between caching and routing. In this paper, building on top of Q-routing we propose a caching strategy, namely Q-caching, which leverages information that is already collected by the routing algorithm. Q-caching promotes content diversity in the network, reducing the load at custodians and average download times for clients. In stylized topologies, we show that the gains of Q-caching against state-of-the-art algorithms are significant. We then consider the RNP topology, and show that Q-caching performance is more flexible while competitive when compared against existing algorithms.*

1. Introduction

Information-centric networking (ICN) is a new networking paradigm that shifts the basic network service semantics from *delivering the packet to a given destination address* to *fetching data identified by a given name* [Afanasyev et al. 2014, Jacobson et al. 2009]. In the ICN paradigm, routers are equipped with caches, with the aim of increasing network performance and scalability. As the network service is content-aware, users can opportunistically download replicas of the content which are closer to the requesters.

In traditional IP networks when there is a change in the network, routing protocols need to exchange routing updates in order to maintain the topology consistent. Thus, IP routing protocols need to converge fast in order to keep packet delivery after network changes. Fortunately, unless a failure occurs, IP networks are typically static in short timescales.

On the other hand, ICN is extremely dynamic because it is a highly distributed caching network, with content location and availability changing over time. In this case, even a static network will have dynamic properties, as the location of content may change due to caching. These differences made [Yi et al. 2014] rethink the role of routing in ICN.

*Now at Pontifícia Universidade Católica do Rio de Janeiro (PUC-RIO), Rua Marquês de São Vicente, 225, Gávea - Rio de Janeiro, RJ - Brasil - 22451-900

Q-routing [Boyan and Littman 1994] was proposed to address the problem of packet routing in dynamically changing networks. It uses single-hop delays to continually estimate the cost to retrieve a packet, called cost-to-go, essentially implementing an asynchronous version of the Bellman-Ford shortest paths algorithm using only local information. It is asynchronous because each router uses only local informational without any route update message.

In recent work, [Chiocchetti et al. 2013] propose a mechanism called *Inform* which consists in extend Q-routing to work in information-centric networks, by estimating not the best routing decision for a particular destination, but for a particular content.

In this paper, we propose a reinforcement learning based solution to the joint problem of content placement and routing, which naturally embraces the notion of utility for the two decisions. This is motivated by the fact that leveraging advantages of ubiquitous caching requires coordination between content placement (insertion and eviction decisions) and content search (routing decisions) [Rossini and Rossi 2014]. In particular, we exploit the fact that Q-routing computes the *cost-to-go* and use it to make not only routing but also caching decisions in a weighted least-frequently used (WLFU) manner, evicting the item with the *minimum expected cost* (MEC) to retrieve. Then, Q-routing and MEC are combined in order to efficiently route requests and store content with the goal of minimizing the download time experienced by users. The resulting combination is referred to as Q-caching.

With this kind of approach, the routing and caching decisions therefore influence each other, as caching an item will cause the routes to change, and changing a route changes the requests seen by other nodes in the network, changing their caches. It is therefore important for the two to be well-coordinated, which we realize by basing them on the same information.

There have been separate efforts to incorporate the concept of utility for placement (e.g., web proxy caching [Wooster and Abrams 1997] and TTL caches [Dehghan et al. 2016]) and search decisions (e.g., Q-routing [Boyan and Littman 1994]). Nonetheless, to the best of our knowledge none of the prior works accounted for the notion of utility in an integrated manner for placement and routing decisions.

The paper is organized as follows. First, we introduce Q-routing for information-centric networking in Section 2. Then, we propose our cost-to-go based caching strategy in Section 3. In Sections 4 and 5 we introduce the local cost minimization problem and some of the key features of Q-caching. Section 6 describes the workload used in our trace-driven experiments. We will continue by presenting our preliminary results, related work and discussions in Sections 7, 8 and 9, and drawing conclusions in Section 10.

2. Q-routing

In Q-routing [Boyan and Littman 1994], the cost-to-go is estimated locally and asynchronously using reinforcement learning [Sutton and Barto 1998]. It uses a tabular *value function* $Q_x(d, y)$ which stores, for every node x, the cost to reach destination d when routing through immediate hop y. In the original paper d is a destination *node*, but in ICNs it is a content.

The Q table is estimated on-line via the Q-learning update rule:

$$Q_x(d,y) \leftarrow Q_x(d,y) + \alpha\left(t + \left(\min_{y' \in \mathcal{N}(y)} Q_y(d,y')\right) - Q_x(d,y)\right) \quad (1)$$

where α is the parameter of an exponential moving average filter, $\mathcal{N}(y)$ is the set of immediate neighbors of node y and t is the cost (e.g., delay) between x and y. The minimum operation in Equation 1 is performed by node y when it receives the request from node x, and returned as part of the acknowledgement.

When routing a request, the Q table is consulted to find the neighbor y with the lowest cost-to-go for a particular content d. Although in general some form of *exploration* is necessary to find optimal routes (by randomly selecting interfaces which are not currently thought to provide the lowest cost), in practice the algorithm has shown to perform well [Boyan and Littman 1994]. Alternatively, a separate *exploration phase* [Chiocchetti et al. 2013] may be used to first estimate the Q values.

Note that Equation 1 estimates the expected cost incurred by node x to fetch content d through its neighbor y. Let $u(x,d,y)$ be the instantaneous local utility associated to sending a request for content d from node x to its neighbor y. In general, any utility function may be used in our estimates, by letting $t = -u(x,d,y)$. For example, some content may be treated preferentially, or some nodes may be avoided.

3. Q-caching

In previous work, Q-routing was used to find content in an information centric network using least-recently used (LRU) caching [Chiocchetti et al. 2013]. LRU is a ubiquitous caching strategy due to its fast and easy implementation, combined with good performance. However, the availability of the cost-to-go allows us to make better caching decisions than this simple heuristic. In particular, LRU does not take into account the downstream availability of the content.

If we wish to minimize the path delay for a given request, it makes sense to cache the items that are most difficult to obtain, i.e. have the highest cost-to-go $Q_x(d)$. Given a request distribution $r_x(d)$ that specifies the probability of a request for information d arriving at node x, we wish to minimize the expected cost:

$$\sum_d r_x(d) Q_x(d) = \sum_d r_x(d) \min_{y \in \mathcal{N}(x)} Q_x(d,y) \quad (2)$$

This is achieved by sorting the content according to this expected cost and caching the items with the highest values. If $r_x(d)$ is estimated by request counting, this amounts to a weighted least-frequently used (WLFU) caching policy [Wooster and Abrams 1997], where the weight is the minimum expected cost (MEC) to obtain the content.

Algorithm 1 describes the behavior of a single node. The request distribution is approximated by request counting, on line 4. Then, the caching decision is made. On the backward path, the requested item is added to the cache (line 7). If that causes the cache to overflow, the item with the minimum expected cost to obtain is evicted (line 9). The routing decision is taken in line 10, and the minimum in Equation (1) is returned on acknowledgement (line 12). Finally, the Q table is adjusted according to Equation (1) in line 13.

Algorithm 1 Q-caching

$Q \leftarrow 0, \hat{r} \leftarrow 0$
$C \leftarrow \emptyset$
procedure Q-CACHING(x,d)
    $\hat{r}_x(d) \leftarrow \hat{r}_x(d) + 1$ ▷ Count requests
    if $d \in C(x)$ **then**
        return content to client, starting backward path
    Add d to $C(x)$ ▷ On backward path
    if $|C(x)| > B$ **then**
        Evict $\arg\min_{d' \in C(x)} \left(\hat{r}_x(d') \min_{y' \in \mathcal{N}(x)} Q_x(d', y')\right)$
    $y \leftarrow \min_{y' \in \mathcal{N}(x)} Q_x(d, y')$ ▷ Routing decision
    Route request for d to y, measuring t
    Receive $q = \min_{y' \in \mathcal{N}(y)} Q_y(d, y')$
    $Q_x(d, y) \leftarrow Q_x(d, y) + \alpha\left(t + q - Q_x(d, y)\right)$

4. Cost Minimization

In this section we relate Q-caching to the cost minimization problem. We introduce the local optimization problem in Section 4.1, and then use it to relate Q-caching to other policies in Section 4.2.

4.1. Local Optimization Problem

In this section, we consider the local optimization problem faced by each cache-router.

variable	description
$h_x(d)$	hit probability
$r_x(d)$	probability that request is for content d
λ_x	total request arrival rate (requests/s)
$\lambda_x(d)$	request arrival rate for content d, $\lambda_x(d) = \lambda_x r_x(d)$ (requests/s)
$E[D_x]$	expected delay experienced by node x
$Q_x(d, y)$	cost to go (computed using Q-routing)
$Q_x(d)$	minimum cost to go (computed using Q-routing)
B	buffer size
$w_x(d)$	weight equal to $r_x(d)Q_x(d)$

Table 1. Notation

All quantities in this section account for a single tagged cache x. Let λ_x be the request arrival rate. Let $\lambda_x(d)$ be the request rate for content d. Then, $\lambda_x(d) = \lambda_x r_x(d)$, where $r_x(d)$ is the fraction of requests for content d. Let $h_x(d)$ be the hit probability of content d at cache x, i.e., $h_x(d)$ is the probability that content d is stored at cache x. The average delay to download a typical content at cache x is $E[D_x] = \sum_d r_x(d)(1 - h_x(d))Q_x(d)$. A summary of the notation used throughout this paper is presented in Table 1.

The optimization problem faced at cache-router x is,

$$\text{maximize} \quad \sum_d r_x(d) h_x(d) Q_x(d) \tag{3}$$

$$\text{subject to} \quad \sum_d h_x(d) = B \tag{4}$$

Note that, as in [Melazzi et al. 2014], we consider an expected buffer size constraint, i.e., the number of expected items in the cache cannot exceed the buffer size B.

Let $w_x(d) = r_x(d)Q_x(d)$. The weight $w_x(d)$ is approximated online as the product of the number of request counts to a content multiplied by the cost to go. The optimal solution to the problem above consists of storing the contents that have larger values of $w_x(d)$, i.e., setting $h_x(d) = 1$ for the contents with larger values of $w_x(d)$ and $h_x(d) = 0$ otherwise. This motivates the caching policy proposed in this paper, which consists of storing the items with largest weights (i.e., largest costs $w_x(d)$) and evicting those with *minimum expected cost* (MEC).

4.2. Contrasting MEC Against Other Policies

The optimal solution of (3)-(4) provides insight about how MEC compares against pre-existing policies. If $Q_x(d) = 1$, for all d, we note that MEC degenerates to storing only the most popular items. This intuitive policy has formally shown to be optimal in [Liu et al. 1997].

Consider now the case where the popularity of the items is not known beforehand. In that setup, a cache which does not take into account the topology and costs of the remainder of the network when making its eviction decision can use LRU or LFU as proxies to the strategy of storing only the most popular items.

MEC, in contrast, accounts for the topology of the remainder of the network when making its eviction decisions. While doing so, each cache 1) assumes that the network is optimized and 2) does not account for the impact of local decisions on the state of the other nodes in the network. Together, properties 1) and 2) provide a simple and distributed strategy, which however is suboptimal in general. The optimal routing and placement problem, accounting for all cache interdependencies, can be shown to be NP-hard [Neves et al. 2010].

5. Q-caching Features

Next, we discuss some of the features derived by combining Q-routing with MEC.

5.1. Utility-Driven Caching

Q-routing naturally allows for utility-driven routing. By leveraging the cost-to-go computed by Q-routing to drive caching decisions, we also allow for utility-driven decisions for content storage. In particular, this provides additional flexibility with respect to existing policies such as LFU or LRU, whose utility is essentially coupled to the content hit rate.

Note that LRU automatically stores all content that passes through a given cache. Under Q-caching, in contrast, the controller might decide not to store a given content which passes through the cache. This, in turn, may lead to less churn.

5.2. Enabling Content Diversity

Q-caching promotes content diversity in the system, by favoring the storage of different contents at different caches. Under LRU, the most frequently requested contents are always locally stored, irrespectively of the state of their neighbors. The state of the neighbors affects a cache only through their miss streams. Under Q-caching, in contrast, the state of the neighbors directly impacts the caching decisions at a given tagged cache. This is because the cost-to-go is distributedly computed taking into account the distribution of the content in the network. One of the consequences of increased content diversity is reduced load at the custodians, in addition to reduced expected download times.

5.3. Handling Loops

One of the critical aspects of any routing algorithm relates to the handling of loops. Initial work on ICN suggested that Pending Interest Tables (PITs) could prevent loops [Dai et al. 2012, Jacobson et al. 2009]. This is because PITs associate to each pending request a random *nonce*, which can be used to discard duplicate requests. However, recent work [G-L-Aceves 2015] indicates that ICNs are subject to routing loop problems even when cache-routers are equipped with PITs.

Routing in ICN can rely on Distance-Vector protocols, like Bellman-Ford, or Link-State Protocols, like Dijkstra. While the Dijkstra algorithm gracefully handles positive loops, Bellman-Ford based algorithms may lead to infinite relaying under adverse conditions. As Q-routing is based on Bellman-Ford, it is subject to the *count-to-infinity* problem, i.e., the convergence might take significant time in the worst case.

The *count-to-infinity* problem is exacerbated by the continuing variation of content availability due to caching. In this paper, we deal with this by not making the Q updates immediately influence the routing decisions. Instead, the Q-table used for routing is updated only periodically, allowing the Q values to settle in between routing updates. Using such a separate "target" Q-table has proven to be a stabilizing factor in recent work on large-scale reinforcement learning [Mnih 2015]. We leave a more theoretical study of the convergence to future work.

5.4. Complexity

The majority of the computational effort is expended on the caching decision, as it requires evaluating the utility of all cached objects $(O(B|\mathcal{N}(x)|))$. However, as the utility only changes when the Q value or popularity changes, the caching decision could use a priority queue, leading to a scalable $O(log(B) + |\mathcal{N}(x)|)$ implementation. The periodic update only involves memory copy, and is therefore not very expensive.

6. Workload

Next, we present the workload used in our trace-driven numerical evaluations. The physical topology is obtained from the Brazilian National Research Network (RNP). [1] The number of clients at each point of presence (PoP) is assumed to be proportional to the number of students of that region given by the 2013 university students census [INEP 2013].

[1]The topology is obtained from `http://web.archive.org/web/20140905111428/http://www.rnp.br/servicos/conectividade/rede-ipe`

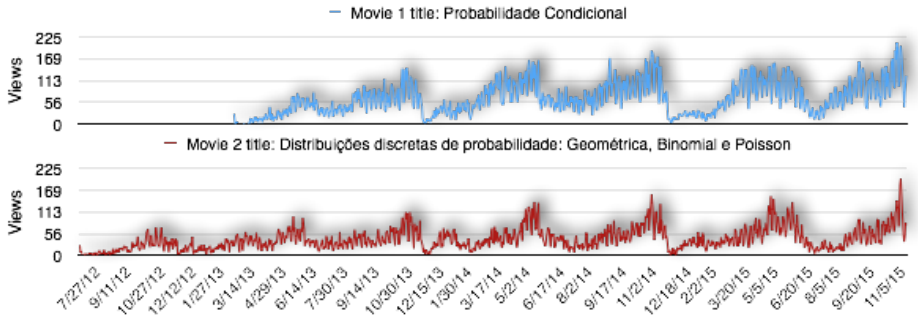


Figure 1. Time series associated to the top two most popular movies

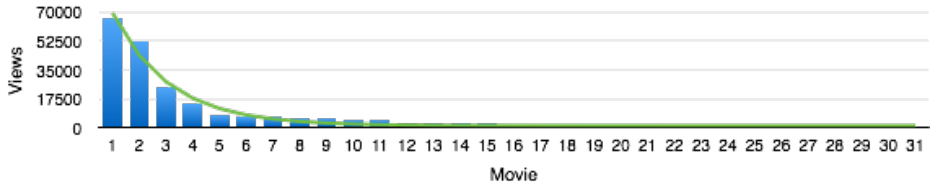


Figure 2. Distribution of the number of views per movie

We consider a content catalog comprising 31 Youtube videos from the UFRJ Performance Evaluation channel. The videos were published between 2012 and 2015, and count with up to 200,000 views and up to 700 followers. Figure 1 illustrates the time series of the two most popular movies. Each series start at the movie publishing date. As a side contribution of this work, we make available the collected traces [UFRJ 2015]. Among the interesting properties extracted from the data, we point out the following (Figure 1):

Periodicity: We note the periodic behavior of the number of views. During the semester, the number of views grows. During vacations, it decreases. During working days, the number of views is usually smaller than during the weekends. The trend is consistent among the movies.

Non-stationarity: As a general trend, the number of views increases from an year to the other. In our trace-driven simulations we consider non-stationary workloads. The simulation is divided into epochs, where each epoch corresponds to a day in the trace. The arrival rate for each content at each day is assumed to be proportional to the number of requests issued for that content, which is obtained from the trace.

Figure 2 shows the distribution of the accumulated number of views by November of 2015. The bars correspond to the collected data, and the line is result of curve fitting. The distribution popularity is well approximated by an exponential distribution, with $y = 1473 + 108947\exp(-0.4707x)$. The root mean square of the residuals equals 2393. As a comparison, the minimum root mean square of the residuals obtained with a power law distribution equals 4024.

7. Experiments

Next, we report the numerical results of our experiments. The major strategy combinations considered are shown in Table 2. We consider four algorithms (Q-LRU, Q-routing, Q-caching and SPF + LRU), three caching strategies (LRU, LFU and MEC) and two routing strategies (shortest path first routing (SPF) and Q-routing). Q-LRU is a simplified version of *Inform*. Under SPF, requests take the path with minimum cost towards the custodian, assumed to be fixed and given. In all cases, if a replica is opportunisti-

cally found, the request is immediately served. We first consider simple topologies in Section 7.1. Then, we contrast different solutions under the RNP topology in Section 7.2.

Algorithm	Routing	Caching	Notes
Q-LRU	Q-routing	LRU	Simplified Inform. LRU is too volatile to work with Q-routing
Q-caching	Q-routing	MEC	less volatile than Inform, comparable to Q-LFU
Q-LFU	Q-routing	LFU	comparable to Q-caching
SPF+LRU	shortest path first	LRU	see Section 7.2

Table 2. Some of the strategy combinations considered in this paper

In our simulations, run in Matlab, we assume stationary latency between hops, with zero download delays (ZDD assumption) on the backward path, which is assumed to be identical to the forward path. As such, when a piece of content is found, all caches on its request path are immediately updated in accordance to the ZDD assumption. All nodes run in lock-step, and process their entire request queue on every step. The ability of Q-routing to dynamically route around congestion is therefore not used, but will be the subject of future work with more detailed simulations.

7.1. Simple Topologies

Figure 3 presents the three simple topologies considered in our experiments. The layered topology (Figure 3(a)) is inspired by the Akamai network [Sitaraman et al. 2014]. The tree topology (Figure 3(b)) is used to clarify the role of caching as opposed to routing in network performance.

The reference setup considered in our experiments is shown in Table 2. The request arrival rate for the k-th content is given by $1/k^{\alpha}$, for $k = 1, \ldots, C$, where α is referred to as the *Zipf parameter*. Parameters are varied according to our experimental goals. For each parameter setting we simulated 10 runs to obtain the mean and the 95% confidence interval of the metrics of interest, represented by a line and a shaded region in the plots that follow, respectively.

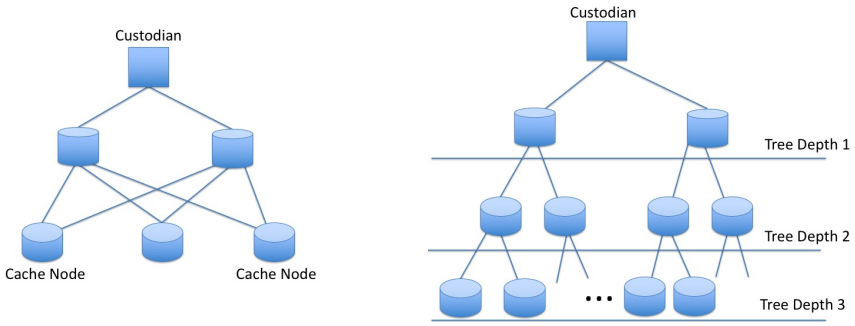


Figure 3. Hierarchical topologies: (a) layered; (b) tree.

Number of nodes, N	Cache size, B	Zipf parameter, α	Topology	Number of files, C	Custodian cost	Exploration rate	Link cost
10	10	0.8	layered	100	100	0.05	1

Table 3. Reference setup for experiments

7.1.1. Optimal Exploration Rate

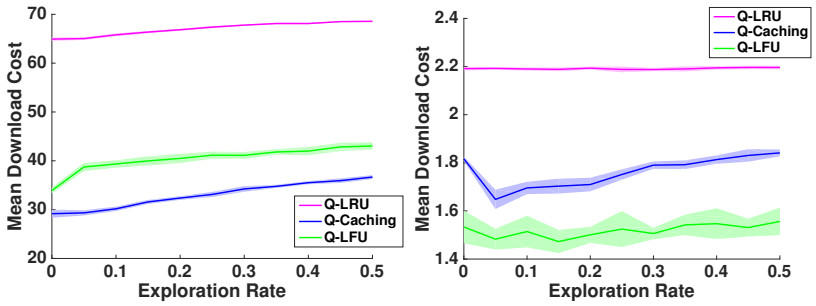
Next, we study the impact of the exploration rate on the metrics of interest. To this aim, we consider a network of 15 nodes, and vary the exploration rate. The other parameters are shown in Table 3.

We consider a layered topology with exogenous arrivals occuring at every node. Recall that exploration is necessary under Q-routing in order to learn the distribution of cost-to-go. This cost-to-go is used for routing decisions by Q-routing and for caching decisions by Q-caching, i.e., when caching is performed using LFU or LRU, the cost-to-go is used only for routing decisions.

According to Figure 4(a), when the custodian cost is 100, the optimal exploration rate is 0 and 0.05 for Q-caching and 0 for Q-LFU. When the custodian cost is 1, the optimal exploration rate is roughly 0.05 for Q-caching and 0.15 for Q-LFU as shown in Figure 4(b). These results shows that is worth to do a little *exploration* when all links has similar costs. When there are different costs, is better *exploit* the optimal routes. This happens because different costs implies in more *stable* topologies.

Figure 4(a) also shows that Q-caching outperforms Q-LFU when the custodian cost is 100, while Figure 4(b) shows the opposite when the custodian cost is one. This means that the Q-caching is *cost-aware* as shown in Section 7.1.3.

Note that Q-LRU is not sensitive to the exploration rate. We believe that the estimates of the cost-to-go are not very helpful under LRU due to the high content churn associated to such a policy, which causes high variability in the distribution of items in the caches over time. This variability precludes the quick convergence and gains of Q-routing.



(a) Custodian Cost=100 (b) Custodian Cost=1

Figure 4. Impact of exploration rate on the Mean Download Cost for two differents custodian costs

7.1.2. Q-caching Increases Space Diversity

Next, we indicate that Q-caching increases space diversity, and as a consequence, decreases mean download cost. To this aim, we consider a tree topology with l levels and $2^{l+1} - 1$ nodes. The remaining parameters are shown in Table 3. Figure 5 shows how the mean download cost varies as a function of the tree depth. As the tree depth increases, the mean download cost decreases more significantly under Q-caching as opposed to other strategies. Because Q-caching considers cost as well as popularity, the state of the neighbors impacts caching decisions. Thus Q-caching might not store a popular content if it is

easy to retrieve from a neighbor. This increases space diversity because different replicas tends to be distributed all over the network, allowing requests to opportunistically download content from within the network and avoiding requests to the custodian, assumed to be associated to the most costly link in the system.

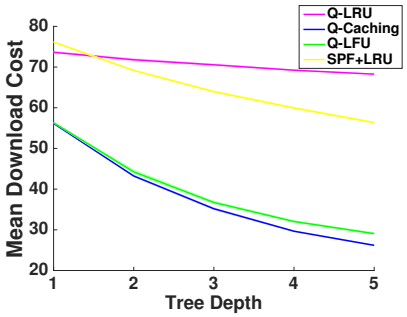
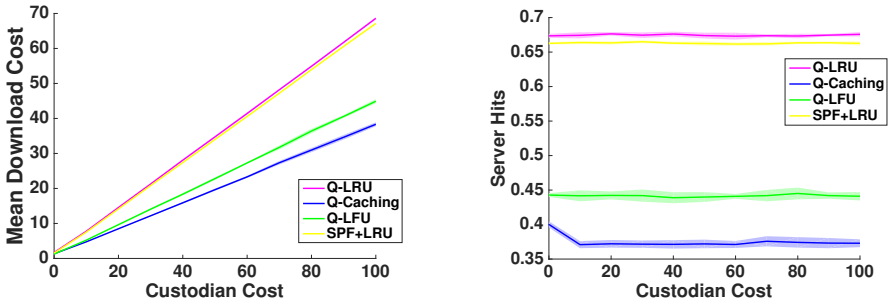


Figure 5. Impact of tree depth on mean download time

7.1.3. MEC is Cost-aware

Figure 6(a) shows the impact of the costs of accesses to the custodian on the mean download cost. We consider the reference setup of Table 3, with custodian cost varying between 0 and 100. Although the mean download cost increases linearly with respect to the cost for all the strategies considered, the slope of Q-caching is smaller than its counterparts. This indicates that Q-caching, by being cost-aware, can gracefully handle changes in custodian costs. Figure 6(b) shows how the number of server hits varies as a function of the custodian cost. While Q-caching adjusts itself to account for cost changes, the other strategies considered are cost-oblivious and maintain the same server hit rate irrespective of server costs.



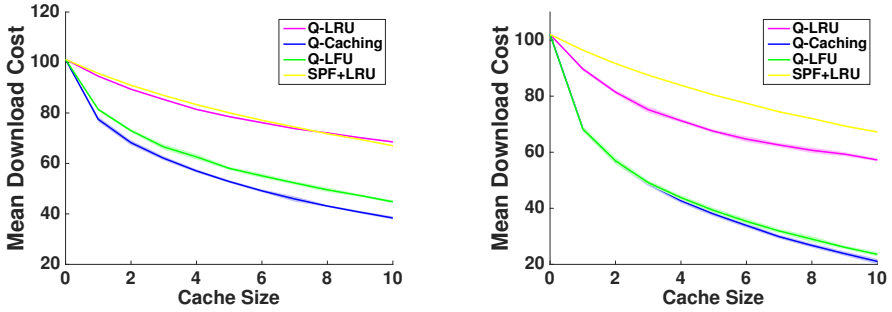
(a) (b)

Figure 6. Impact of custodian cost

7.1.4. Impact of Request Locality

Next, our goal is to study the impact of request locality on mean download cost. We consider the reference setup shown in Table 3. In Figures 7(a) and 7(b) we consider the cases where exogenous requests arrive at every node and only at the leaves, respectively. As Figure 7(a) shows, Q-caching is better than Q-LRU, and slight better than Q-LFU, when there are exogenous arrivals at every node. In contrast, Figure 7(b) shows that Q-LFU is slightly better than Q-caching when there are exogenous arrivals only at the edge. This is because, due to symmetry, the cost-to-go associated to items which are not

stored in the leaves is roughly the same for all nodes. Therefore, the gains of Q-caching due to heterogenous cost-to-go values are not leveraged.



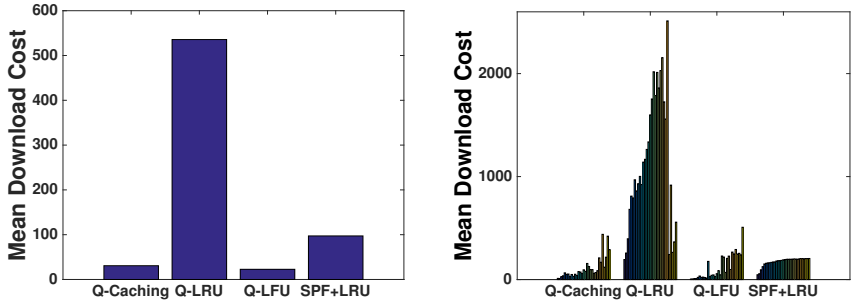
(a) arrivals at every node (b) arrivals only at the edge (leaves)

Figure 7. Impact of request locality

7.2. RNP Topology and Youtube Traces

To validate the strategies proposed, real network topology and video traces are used as described in Section 6. The video custodian is placed at João Pessoa. Every point of presence of the RNP network is associated to a cache-router with capacity to store up to three videos. All videos have unit size. The cost of a link is inversely proportional to its bandwidth. We set 20 Gbps as the reference bandwidth, meaning that its cost is normalized to 1. Henceforth, we consider only normalized costs, referred to simply as *costs*. The custodian has a link with capacity of 100 Mbps, i.e., its cost is 200.

Figures 8(a) and (b) show the results obtained using the RNP topology and the Youtube trace-driven simulations. Figure 8(a) shows the mean download cost to retrieve all files for the four considered strategies. As the figure indicates, Q-LFU is slightly better than Q-caching, and Q-LRU is the worst strategy. Figure 8(b) shows the mean download cost for each video. As in Figure 2, videos are sorted according to the total number of views. Under Q-LRU, the mean download cost is high for all videos. Under Q-caching and Q-LFU, the popular videos have smaller minimum download costs. Under Q-caching, 26 videos have mean download costs lower than 200, in agreement with the hypothesis that Q-caching increases space diversity. Under Q-LFU, in contrast, only 20 videos have mean download cost lower than 200. Under SPF+LRU, the four most popular videos have mean download costs lower than 200. The mean download cost of all other videos is roughly equal to 200.



(a) averaged over all files (b) per file

Figure 8. Mean download costs (RNP topology and Youtube video traces)

Figure 9 shows the evolution of the mean download cost over time for the four considered algorithms. This figure shows although the mean download cost of Q-LRU widely varied during the simulation period, SPF+LRU showed a better and more stable behavior. As previously discussed, this occurs because LRU yields high content volatility in the caches, not allowing Q-routing to properly converge. Under LRU, every new request to a content that is not cached leads to an eviction, naturally producing more churn, which needs to be coped by Q-routing. When Q-routing is combined with MEC (Q-caching) and LFU (Q-LFU), in contrast, a popular content might never show up in the miss stream. This feature promotes stabilization.

Note that the convergence of Q-caching is slower than that of Q-LFU. Nonetheless, after convergence the mean download costs of Q-caching and Q-LFU are 4 and 10, respectively. This means that depending on the dynamics of the content catalog and of the popularity of contents, Q-caching might be preferred over Q-LFU.

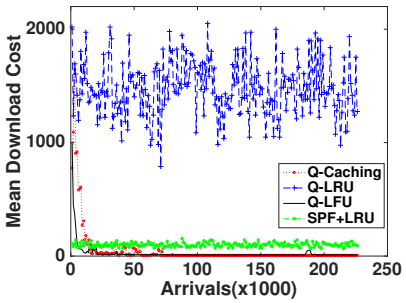


Figure 9. Evolution of the mean download cost over time

8. Related Work

The role of routing and the interplay between routing and caching in ICNs is discussed in [Yi et al. 2014, Chiocchetti et al. 2012, Chiocchetti et al. 2013, Rossini and Rossi 2014]. As such previous work suggests, jointly solving the routing and caching problems might lead to significant performance gains. However, the joint analysis of these two aspects is non-trivial, and taking them into account together in a simple manner was still an open problem. In this paper, we propose to use the cost-to-go provided by Q-routing for caching decisions, integrating routing and caching in a simple way.

When determining the steady state occupancy of the caches, one approach consists of searching for a fixed-point solution which is compatible with the routing and caching decisions [Tortelli et al. 2016]. We believe that the solutions proposed in this paper are also amenable to analysis through adaptations of the models proposed in [Rosensweig et al. 2010].

Efficient content placement in ICN with TTL-caches has been studied in [Domingues et al. 2013]. [Dabirmoghaddam et al. 2014, Fayazbakhsh et al. 2013] argued that *edge-caching* produces most of the gains of *everywhere-caching*. Our results suggest that Q-caching benefits from everywhere-caching because it increases content diversity.

The use of reinforcement learning (RL) for decision making in ICNs has been considered in other works. [Chiocchetti et al. 2012] studied the classical RL tradeoff between exploitation and exploration and then proposed Inform [Chiocchetti et al. 2013], which combines Q-routing with LRU for improved performance. [Bastos et al. 2015] also propose the use of RL for routing decisions. In this work, in contrast, we propose the use of RL for routing and caching decisions, in an integrated fashion. To the best of

our knowledge this paper is the first to leverage information provided by reinforcement-learning based routing algorithms for caching decisions.

9. Discussion

Next, we discuss some of the simplifying assumptions considered in this work, together with possible directions for future work.

Forward and Backward Routes: In this paper, we focused on the forward (upstream) routes, namely the routes between requestors and custodians. We leveraged the opportunistic encounters between requests and replicas of the content closer to users. The backward (downstream) routes were assumed to be the same as the forward routes, and the zero download delay (ZDD) assumption was considered. If backward routes may be distinct from forward routes, additional flexibility might be gained to determine how to route the content to users, so as to strategically place new replicas where needed.

Routing and Caching Utilities: We assumed that the utilities associated to routing and caching decisions are the same. However, in general they might be associated to distinct utilities. In such case, additional control messages might be required. The gains, however, might compensate such overhead specially if routing costs (e.g., due to congestion) are very distinct from content costs (e.g., due to the availability of custodians for certain rare or unpopular content).

10. Conclusion

Caching and routing are two of the building blocks of ICNs. In this paper, we showed how to leverage information provided by Q-routing in order to make caching decisions. The proposed solution, Q-caching, is simple and flexible, allowing for different utilities and metrics of interest to be taken into account when coupling caching and routing for increased performance. We numerically investigated the effectiveness of Q-caching, and contrasted it against different schemes, including Q-routing coupled with LRU (the basis for the state-of-the-art Inform). We have found that in the scenarios investigated Q-caching is either better or competitive againts the considered counterparts. Future work consists of establishing the conditions under which Q-caching is optimal, and studying its convergence properties.

Acknowledgment: This work was supported in part by CNPq, PETROBRAS and CAPES. Wouter Caarls was funded by a Science Without Borders scholarship (CSF/Bolsa Jovem Talento/CAPES).

References

Afanasyev, A., Burke, J., Zhang, L., Claffy, K., Wang, L., Jacobson, V., Crowley, P., Papadopoulos, C., and Zhang, B. (2014). Named Data Networking. *ACM SIGCOMM CCR*, 44(3):66–73.

Bastos, I., Sousa, V., and Moraes, I. (2015). Uma estratégia de encaminhamento de pacotes baseada em aprendizado por reforço para redes orientadas a conteúdo. In *SBRC*.

Boyan, J. A. and Littman, M. L. (1994). Packet routing in dynamically changing networks: A reinforcement learning approach. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 671–678. Morgan-Kaufmann.

Chiocchetti, R., Perino, D., Carofiglio, G., Rossi, D., and Rossini, G. (2013). Inform: a dynamic interest forwarding mechanism for information centric networking. In *ICN*, pages 9–14. ACM.

Chiocchetti, R., Rossi, D., Rossini, G., Carofiglio, G., and Perino, D. (2012). Exploit the known or explore the unknown?: Hamlet-like doubts in ICN. In *ICN*, pages 7–12. ACM.

Dabirmoghaddam, A., Barijough, M. M., and G-L-Aceves, J. (2014). Understanding optimal caching and opportunistic caching at the edge of information-centric networks. In *ICN*, pages 47–56. ACM.

Dai, H., Liu, B., Chen, Y., and Wang, Y. (2012). On pending interest table in named data networking. In *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '12, pages 211–222, New York, NY, USA. ACM.

Dehghan, M., Massoulie, L., Towsley, D., Menasche, D., and Tay, Y. (2016). A utility optimization approach to network cache design. In *INFOCOM*.

Domingues, G. d. M. B., Leão, R. M. M., Mensché, D. S., et al. (2013). Enabling information centric networks through opportunistic search, routing and caching. In *SBRC*.

Fayazbakhsh, S. K., Lin, Y., Tootoonchian, A., Ghodsi, A., Koponen, T., Maggs, B., Ng, K., Sekar, V., and Shenker, S. (2013). Less pain, most of the gain: Incrementally deployable icn. In *ACM SIGCOMM CCR*, volume 43, pages 147–158. ACM.

G-L-Aceves, J. (2015). A fault-tolerant forwarding strategy for interest-based information centric networks. *IFIP Networking 2015*.

INEP (2013). Sinopses estatísticas da educação superior 2013 - graduação `http://portal.inep.gov.br/superior-censosuperior-sinopse`. Accessed December 25, 2015.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *CONEXT*, pages 1–12. ACM.

Liu, Z., Nain, P., Niclausse, N., and Towsley, D. (1997). Static caching of web servers. In *Photonics West'98 Electronic Imaging*, pages 179–190. International Society for Optics and Photonics.

Melazzi, N. B., Bianchi, G., Caponi, a., and Detti, a. (2014). A general, tractable and accurate model for a cascade of LRU caches. *IEEE Communications Letters*, 18(5):877–880.

Mnih, V. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Neves, T., Drummond, L. M., Ochi, L. S., Albuquerque, C., and Uchoa, E. (2010). Solving replica placement and request distribution in content distribution networks. *Electr. Notes in Disc. Math.*, 36:89–96.

Rosensweig, E. J., Kurose, J., and Towsley, D. (2010). Approximate models for general cache networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

Rossini, G. and Rossi, D. (2014). Coupling caching and forwarding: Benefits, analysis, and implementation. In *ICN*, pages 127–136. ACM.

Sitaraman, R. K., Kasbekar, M., Lichtenstein, W., and Jain, M. (2014). Overlay networks: An akamai perspective. *Advanced Content Delivery, Streaming, and Cloud Services.*

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Tortelli, M., Rossi, D., and Leonardi, E. (2016). Model-graft: Accurate, scalable and flexible analysis of cache networks.

UFRJ (2015). Youtube traces of performance evaluation channel: `https://www.youtube.com/user/ADUFRJ20121`. Traces available here: `https://goo.gl/NwHf0g`.

Wooster, R. P. and Abrams, M. (1997). Proxy caching that estimates page load delays. In *WWW*, pages 977–986, Essex, UK. Elsevier Science Publishers Ltd.

Yi, C., Abraham, J., Afanasyev, A., Wang, L., Zhang, B., and Zhang, L. (2014). On the role of routing in named data networking. In *ICN*, pages 27–36. ACM.

Uma Estratégia de Cache baseada em Múltiplas Métricas para Redes ICN

Igor Carvalho[1], **Airton Ishimori**[1], **Antônio Abelém**[1]

[1]Grupo de Estudos em Redes e Comunicação Multimídia – GERCOM
Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

{icarvalho, airton, abelem}@ufpa.br

Abstract. *Information Centric Networks (ICN) has been a new network paradigm that has been extensively researched recently. In networking-caching enables routers to cache content in the request path, in which they use a caching strategy to decide whether to store a content or not. However, most of the current strategies are only-one-criterion based in which, due to the dynamics of the network, may not be suitable, resulting in both low network hit ratio and performance. This way, this paper proposes a Multi-Criteria Caching Decision (MCCD) strategy for ICN, which considers three metrics to select the target nodes for caching contents and better distribute them over the network. Our results got a much higher cache hit ratio than the other evaluated strategies and a reasonable download time.*

Resumo. *As Redes Centradas em Conteúdo (Information Centric Networks - ICN) são um paradigma de arquitetura de rede que atraiu muitas pesquisas nos últimos anos. O cache em rede permite que roteadores possam armazenar tal conteúdo, os quais utilizam-se de uma estratégia para decidir se armazenam-no ou não. Porém, a maioria das estratégias são baseadas em apenas uma métrica o que, levando em consideração a dinamicidade da rede, pode resultar em uma baixa taxa de acertos e desempenho da rede. Desta forma, este artigo propõe uma estratégia de decisão de cache baseado em múltiplas métricas para redes ICN, no qual usamos três métricas para selecionar os nós alvos e, com isso, melhor distribuir o conteúdo na rede. Nossos resultados obtiveram uma taxa de acertos bem maior que as outras propostas avaliadas e um tempo de download razoável.*

1. Introdução

Na última década, as Redes Centrada no Conteúdo (*Information Centric Networks - ICN*) surgiram como uma das arquiteturas mais promissoras no contexto da Internet [Ahlgren et al. 2012]. Neste modelo, a obtenção de conteúdo dá-se por meio de requisições baseadas nos nomes de tais conteúdos e não por meio de identificação usando o endereço IP. Quando usuários solicitam conteúdos, uma rota é estabelecida até o servidor ou provedor de conteúdo e os nós intermediários podem armazená-los em *cache*. Isto permite que novas requisições possam ser atendidas por tais nós, reduzindo tanto a sobrecarga no servidor de conteúdo como o consumo de largura de banda, aumentando sua disponibilidade na rede [Dannewitz 2009].

De acordo com um estudo feito pela Cisco, o tráfego IP global crecerá a uma taxa anual de 23% entre 2014 e 2019. Até 2019, o tráfego total de vídeo, somando usuários comuns e empresas, será 77% de todo o tráfego da Internet [Cisco 2015]. Desta forma, uma estratégia de decisão de *cache* influencia fortemente no desempenho da rede, no sentindo de escolher os nós alvos para atender as solicitações dos usuários. Caso ela não atenda às requisições razoavelmente, consequentemente haverá uma baixa taxa de acertos *hit ratio*), resultando em alta carga no servidor e longos atrasos [Zhang et al. 2013].

A estratégia padrão em ICN é a LCE (*Leave a Copy Everywhere*). Ela armazena o conteúdo solicitado em todos os nós entre um nó solicitante e um nó que possui tal conteúdo, podendo ser um *cache* intermediário ou o servidor [Jacobson et al. 2009].Ela garante uma rápida difusão do conteúdo por toda a rede. Porém, algumas questões sobre seu desempenho foram levantadas. A primeira é sobre sua alta redundância, isto é, o mesmo conteúdo é desnecessariamente copiado em todos os nós no caminho reverso da requisição, reduzindo a variabilidade de conteúdo na rede. A outra problemática levantada é que, como o espaço reservado para *cache* dos nós é relativamente menor que o total de conteúdo disponível na rede, o LCE causa uma alta taxa de erros devido às trocas de conteúdo no *cache* do roteador [de Brito et al. 2012].

Dentro do paradigma ICN, há muitos problemas relacionados ao *cache* em rede. De acordo com [Kutscher et al. 2014], dois deles são mencionados como o problema de alocação de cache e distribuição de conteúdo. O primeiro fornece algumas características necessárias pelos nós da rede a fim de que o uso de *cache* na rede seja possível. Elas são: a centralidade dos nós, o tamanho do domínio envolvido, evitar links custosos ou inter-domínios, padrões de tráfego e localizaçao espacial dos usuários em uma parte específica na rede. O segundo está relacionado com a distribuição de conteúdo nos caches, que pode armazenar os conteúdos na rota estabelecida entre o cliente e o servidor (*on-path caching*) ou armazená-los em outros nós localizados em uma outras parte da rede (*off-path-caching*).

Muitas pesquisas sobre cache em redes foram desenvolvidas no contexto de estratégias de decisão de cache na última década. Algumas consideram a quantidade de espaço disponível para cache dos nós [Lee et al. 2013]; uma outra foi desenvolvida baseado num modelo probabilístico que considera a capacidade de cache do caminho entre o nó solicitante e o nó que possui o conteúdo [Psaras et al. 2012]; a centralidade de intermediação dos nós foi considerada por [Chai et al. 2012], isto é, a importância do nó na rede. Observa-se que cada uma delas utiliza uma métrica específica como distância, capacidade de armazenamento dos nós/caminho ou ainda características topológicas. Entretanto, como a rede e as requisições de conteúdo apresentam um comportamento dinâmico, trabalhar com uma métrica para a escolha dos possíveis nós alvos pode fazer que o conteúdo seja armazenado em poucos nós que casem com tal métrica, impedindo uma melhor distribuição do conteúdo pela rede.

Este artigo propõe uma nova estratégia de decisão de cache baseada em múltiplas métricas para redes ICN chamada MCCD - *Multi-Criteria Caching Decision*. Ela tem como base as seguintes métricas para a tomada de decisão: *centralidade de intermediação dos nós*, *tamanho de cache disponível* e a *taxa de acertos de conteúdo do nó* (Explicaremos detalhadamente cada uma delas na seção 3). Partimos da premissa de que se uma estratégia de decisão possui mais de uma métrica como base para decidir onde armazenar

os conteúdos, pode-se aumentar a taxa de acertos de conteúdos geral da rede, dado que o conteúdo é melhor distribuído pela rede por não concentrá-los em nós específicos que casam com uma métrica apenas. Avaliamos nossa proposta e comparamos seu desempenho com outras estratégias. Obtivemos uma taxa de acertos de conteúdos maior do que as demais baseadas em apenas uma métrica e conseguimos um tempo de *download* próximo ou abaixo das demais, dependendo da topologia.

O restante deste artigo está dividido da seguinte maneira: A seção 2 descreve os trabalhos relacionados sobre estratégias de decisão de cache em ICN; a seção 3 descreve a proposta MCCD; a seção 4 comenta sobre a avaliação de desempenho; a seção 5 analisa e discute os resultados obtidos e a seção 6 conclui o artigo e sugere possíveis trabalhos futuros.

2. Trabalhos Relacionados

A proposta *Cache Capacity-Aware CCN* baseia-se no conceito de *capacidade de cache* [Lee et al. 2013]. Este modelo considera a capacidade de armazenamento em cache livre em todos os nós ao longo do caminho de solicitação de conteúdo. Eles criaram uma métrica chamada de *cache capacity value*(ccv), que é a razão entre o tamanho total do cache do nó pelo número de entradas armazenadas em cache. Com isso, um número de ccv é atribuído para todos os nós, bem como o número de saltos que tal nó está do nó solicitante. O nó alvo a armazenar o conteúdo será o que possuir o maior valor de ccv. Os resultados obtidos por essa proposta se mostraram bastante satisfatório e obtiveram uma taxa de acerto 164% maior do que o resultado da segunda melhor proposta. Entretanto, eles não consideraram em sua avaliação variar o tamanho do cache dos nós a fim de ver o comportamento da proposta em cenários com muito pouco ou razoável tamanho de cache disponível para armazenamento.

Uma estratégia de decisão probabilística chamada de *ProbCache* usa como métrica a capacidade de armazenamento do caminho [Psaras et al. 2012]. Ele copia o conteúdo ao longo do caminho estabelecido na requisição com um certo valor de probabilidade baseado na quantidade de espaço de cache disponível naquele caminho. Este valor é inversamente proporcional à distância do nó que armazena o conteúdo. Quanto mais perto este estiver do nó solicitante maior será a probabilidade de armazenamento. Quando o conteúdo é enviado pelo caminho reverso, um valor de probabilidade é gerado aleatoriamente e é comparado com o valor obtido para cada nó naquele caminho. Caso o valor aleatório seja menor que o valor calculado pela proposta, o conteúdo é armazenado em tal nó. Uma vantagem da proposta é que ela pode armazenar o conteúdo mais rapidamente para próximo dos usuários. No entanto, como não considera a capacidade de armazenamentos dos nós individualmente, a estratégia pode escolher um nó alvo com pouco cache disponível e a política de descarte de conteúdo pode apagar o conteúdo rapidamente e uma requisição para este mesmo conteúdo necessite ser enviada para nós mais distantes.

A proposta denominada de *Cache Less for More - CL4M* - é baseada no conceito de centralidade de intermediação dos nós [Chai et al. 2012]. Tal métrica mede a importância dos nós na rede. Ela informa que se um nó está na maioria de dos caminhos mais curtos entre dois outros nós, ele terá uma maior probabilidade de ter um requisição de conteúdo satisfeita, quando comparado com outros nós e, por isso, é estrategicamente

melhor armazenar o conteúdo em tal nó. Desta forma, pode-se aumentar a taxa de acertos de conteúdo geral da rede e diminuir a taxa de descarte.

O trabalho de denominado de *RankCache* propõe o rankeamento dos caches de todos os nós num caminho de requisição de conteúdo [Avelar and Dias]. Para isso, calcula-se número de acertos de conteúdo requisitado cada nó obteve a partir de uma fórmula proposta. Com o resultado do *rank* calculado, um valor aleatório é gerado e comparado com este valor. Caso seja menor que o valor obtido pela proposta, o conteúdo é armazenado em tal nó. Nos testes de desempenho, obteve-se uma taxa de acertos de conteúdo de 10 a 30% melhor que a segunda melhor proposta avaliada.

Sendo assim, foi proposta uma estratégia de decisão de cache baseada em três métricas, as quais são:*centralidade de intermediação dos nós*, *tamanho de cache disponível* e a *taxa de acertos de conteúdo do nó*. Justifica-se o uso dessas três métricas por terem obtido resultados bastante satisfatórios em suas avaliações, além de considerarem importantes características topológicas bem como dos nós. Quando combinadas, acredita-se que tais características podem ainda melhorar o desempenho de nossa estratégia de decisão de cache. Diferentemente das propostas baseadas em apenas uma métrica, nossa proposta consegue melhor distribuir o conteúdo pela rede, já que não o concentra em nós que casam apenas com um tipo de métrica.

3. Proposta baseada em múltiplas métricas - MCCD

A MCCD utiliza três métricas para decidir em quais nós o conteúdo será armazenado, as quais são o valor de centralidade do nó, tamanho de cache disponível e a taxa de acertos. Para isso, a MCCD escolhe o nó com *maior valor de centralidade de intermediação* , *maior tamanho de cache disponível* e *maior número de acertos no caminho da requisição de conteúdo*.

1. *Centralidade de Intermediação do Nó:* A centralidade de *v* é dada pela seguinte fórmula:

$$Centr_v = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

   em que σ_{st} é o total do número de caminhos mais curtos do nó *s* to *t* e $\sigma_{st}(v)$ é o número de caminhos que passam por *v*.
   Porém, calcular a centralidade de todos os nós é muito custoso em termos de processamento e memória, principalmente em redes grandes. Por conta disso, o simulador usado calcula o valor de centralidade de todos os nós à priori e guarda tal informação em cada um deles por padrão.
2. *Capacidade de Cache Disponível:* Seja *m* o tamanho total de cache de um roteador de conteúdo em número de entradas para armazenamento. Seja *c* o total de conteúdo armazenado no roteador em número de entradas. A Capacidade de Cache Disponível (CCD) de um nó *v* é dado pela seguinte equação:

$$CCD_v = m - c$$

   Por padrão, os roteadores de conteúdos guardam tal informação sobre quantas entradas disponíveis há em seus caches e, com isso, a MCCD consegue extrair tal informação ao longo dos nós do caminho de requisição.

3. *Taxa de Acertos de Conteúdo*: Seja R o número de requisições de conteúdo que chegam a um roteador de conteúdo r. Seja H o número de acertos de requisições de conteúdos atendidas por r. Então, a Taxa de Acertos de Conteúdo (TAC) de r é dado por:

$$TAC_r = \frac{H}{R}$$

Conforme detalhado no algoritmo MCCD na figura 1, depois que o caminho de requisição do conteúdo tiver sido calculado, são criadas três listas contendo os nós deste caminho. Para cada lista, verifica-se qual nó possui o maior valor de acordo com o critério considerado. Isto é, dentro do caminho de requisição, o nó com maior valor de centralidade é selecionado numa lista, o nó com maior cache disponível em outra e o nó com maior taxa de acertos na última. Para o cálculo do valor de centralidade, utiliza-se um método que calcula este valor para todos os nós no caminho da requisição e depois é extraído aquele com maior valor. Na segunda lista, faz-se um mapeamento entre o nó e o tamanho do cache disponível. Então, a lista é varrida a fim de encontrar aquele com maior espaço disponível. Semelhantemente, para achar o nó com maior taxa de acertos, é feito um mapeamento dos nós com seus respectivos valores de taxa e encontra-se o nó com maior taxa. Tendo encontrado os nós alvos, o conteúdo volta pelo caminho reverso e então verifica-se se o nó corrente é um nó alvo. Em caso afirmativo, uma cópia do conteúdo é armazenada nele. Caso contrário, os próximos nós são verificados até que todos os nós alvos sejam encontrados.

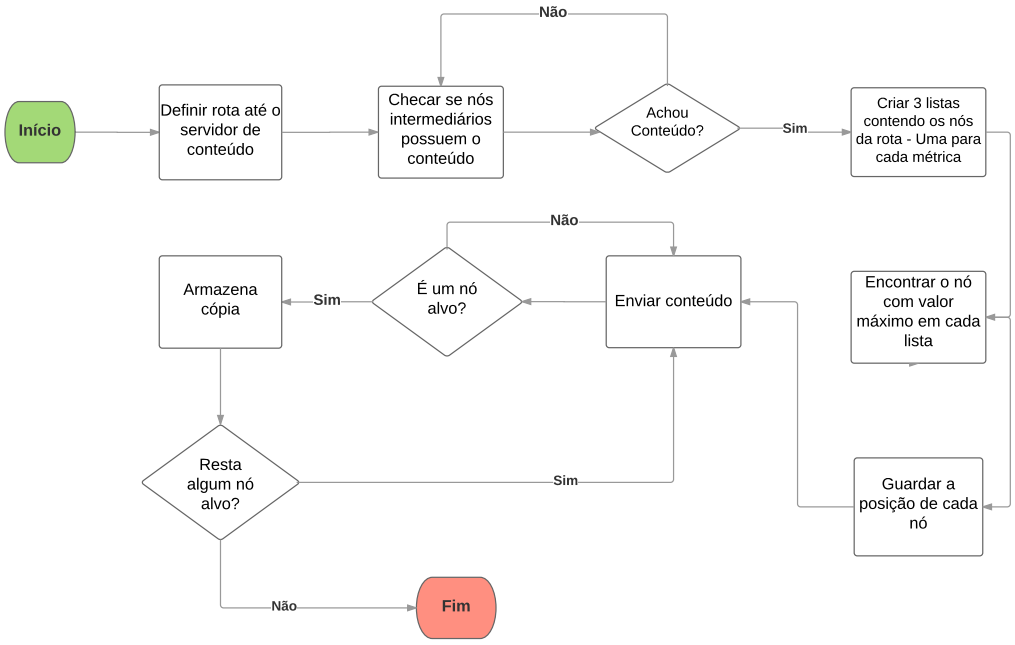


Figura 1. Algoritmo MCCD

Para melhor ilustrar o funcionamento do nosso algoritmo, considere o cenário da figura 2. Assumimos que todos os roteadores (R1 - R11) já possuem conteúdos armazenados em cache, Servidor seja o servidor permanente de conteúdo e os três usuários (1,2 e 3) estejam solicitando conteúdo da rede.

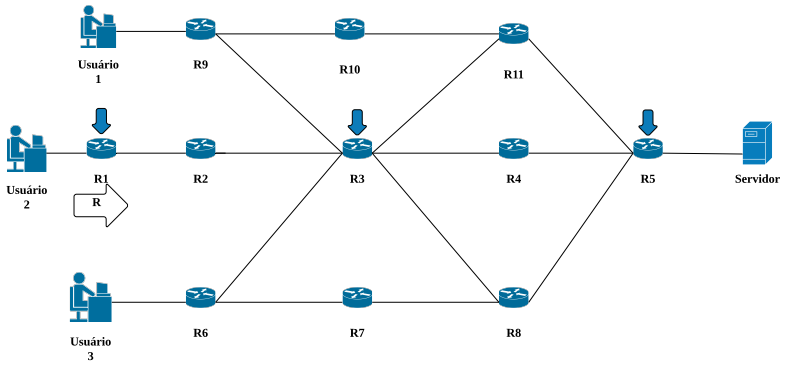


Figura 2. Funcionamento da MCCD

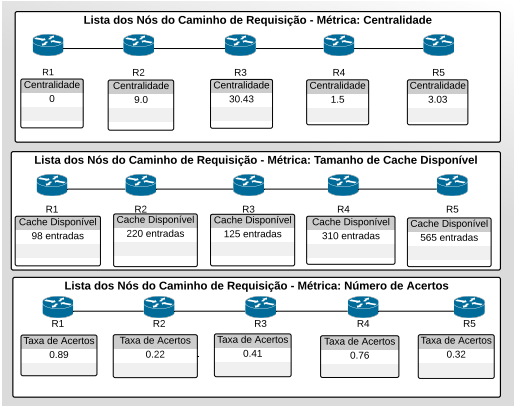


Figura 3. Obtenção das Métricas

Em um dado momento, o usuário 2 envia uma solicitação *R* de um conteúdo *c* para o roteador R1. Este verifica se possui o conteúdo em cache. Se sim, envia-o para o usuário 2. Caso contrário, repassa a *R* para o próximo roteador. Paralelamente, a MCCD vai atualizando o valor das três variáveis a cada salto. Assumimos que *c* não seja encontrado em nenhum roteador intermediário do caminho R1-R5 e *R* chegue até o servidor. Neste momento, a MCCD já selecionou os roteadores que armazenarão o conteúdo quando este voltar pelo caminho reverso. A figura 3 mostra que foram encontrados os roteadores R1, R3 e R5 sendo os nós com *maior taxa de acertos*, *maior centralidade* e *maior espaço de cache disponível*, respectivamente. Caso um nó case com mais de uma métrica, a MCCD evita que o conteúdo seja armazenado novamente neste nó, reduzindo a redundância de cópias na rede.

4. Avaliação de Desempenho

Nossa proposta foi implementada e testada utilizando o simulador Icarus 0.4.0 [9], um simulador de estratégias de de decisão de cache para redes ICN. Este simulador possui algumas implementados e escolhemos alguns deles para comparar seu desempenho e comparar com a nossa proposta.

Tabela 1. Configuração dos parâmetros da simulação

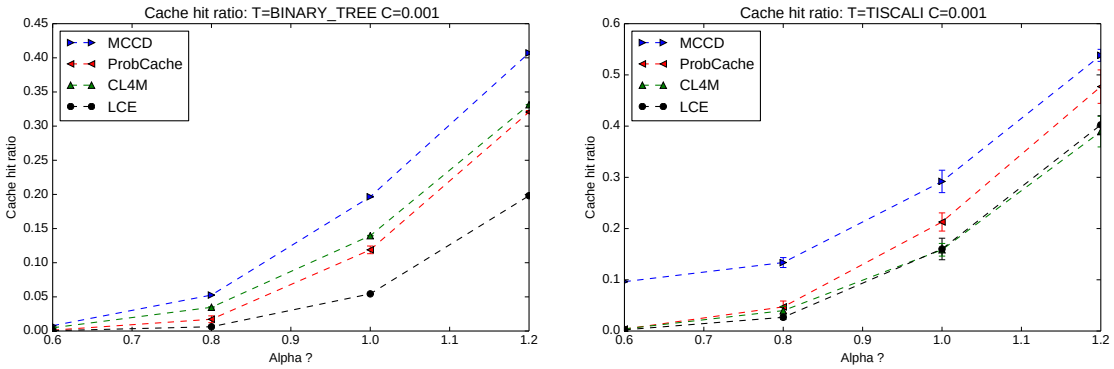
Topologias avaliadas	Árvore Binária e TISCALI
# total de conteúdos no repositório	100.000
Taxa de requisição	100/s
# de requisições para popular caches dos roteadores	100.000
# de requisições avaliadas	200.000
Política de descarte de cache	LRU
Parâmetro Zipf de popularidade de conteúdo	$\alpha = [0.6, 0.8, 1.0, 1.2]$
% de tamanho de cache para os roteadores em relação ao repositório	0.01% 1% e 10%
Estratégias avaliados	MCCD, PROBCACHE, LCE e CL4M
Métricas avaliadas	Taxa de acertos e latência

A tabela 1 mostra os parâmetros de simulação utilizados. As topologias utilizadas foram: Árvore binária com profundidade D = 7. O servidor de conteúdo estava na raiz, os usuários são os nós do último nível da árvore e os nós intermediários são os roteadores de conteúdo. Para a topologia TISCALI, que é um provedor pan-europeu, os clientes são os nós que possuem grau ¡ 4.5. O número de clientes não foi especificado pelos simulador.. Os roteadores possuem grau ¿= 6, resultando em 36 nós; Como servidores foram escolhidos os nós com grau maior que 4.5, resultando em 44 nós. No caso da topologia GEANT, que é um *backbone* europeu, os clientes são nós que possuem grau = 1, resultando em 8 nós; os roteadores são os nós com grau ¿ 2, resultando em 19 nós; os servidores possuem grau igual a 2, totalizando 13 nós. Os 100 K conteúdos foram armazenados nos servidores seguindo uma maneira uniforme. Os conteúdos são tratados como *object* e cada *object* possui 30 vezes o tamanho da sua requisição. O tempo de atraso de links internos é de 2ms e de links externos de 102ms. Foram avaliadas 200.000 requisições de conteúdo e estas requisições seguem um modelo Poisson. A distribuição de popularidade segue um modelo Zipf com um valor de α=[0,6-1,2]. Como tamanho de cache dos roteadores foram configurados três valores: 0.01%, 1% e 10% em relação ao total de conteúdos no repositório; As estratégias avaliados foram a MCCD , PROBCACHE, LCE e CL4M. As métricas avaliadas foram a taxa de acerto, que é definida como a relação entre o número de acertos de conteúdo pelo número total de requisições enviadas, e a latência, que é o tempo necessário para receber o conteúdo solicitado desde sua requisição até o seu recebimento. Todos as avaliações utilizaram um Intervalo de Confiança de 95%.

5. Análise dos Resultados

Primeiramente, analisa-se os resultados de taxa de acertos e depois os resultados do tempo de *download* para as topologias avaliadas. Eles foram analisados por tamanho de cache e, posteriormente, tais resultados serão discutidos mais detalhadamente.

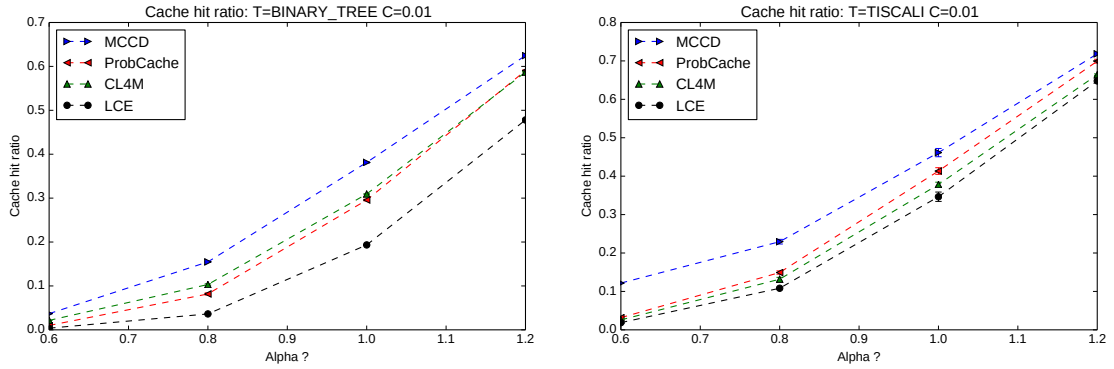
Conforme observa-se na figura 4, para um tamanho de cache de 0.1% a proposta MCCD obteve uma taxa de acerto maior que das outras propostas para ambas as topologias. No teste com árvore binária, (3a), a MCCD conseguiu uma taxa maior para todo valor de alfa, atingindo cerca de 41% de acertos com $\alpha = 1.2$ e uma taxa de acertos 124% melhor que CL4M, que ficou em segundo lugar. Em terceiro lugar ficou PROBCACHE e LCE com o pior resultado. No teste com a topologia TISCALI,(3b), a MCCD obteve uma taxa de acertos bem acima das outras. Com $\alpha = 0.6$, a MCCD obteve uma taxa de 10%, sendo 400% melhor que PROBCACHE para este valor de alfa. Nossa proposta



(a) Árvore Binária **(b)** TISCALI

Figura 4. Taxa de Acertos - Tamanho do Cache = 0.1%

obteve uma certa variação com $\alpha = 1$, mas continuou com um desempenho maior que as demais, conseguindo 53% de taxa de acertos para $\alpha = 1.2$. A proposta PROBCACHE manteve um desempenho similar a CL4M com valores de $\alpha = 0.6 - 0.8$ e superou as outras propostas para valores maiores. CL4M e LCE mantiveram resultados similares e também tiveram uma pequena variação com $\alpha = 1.0$

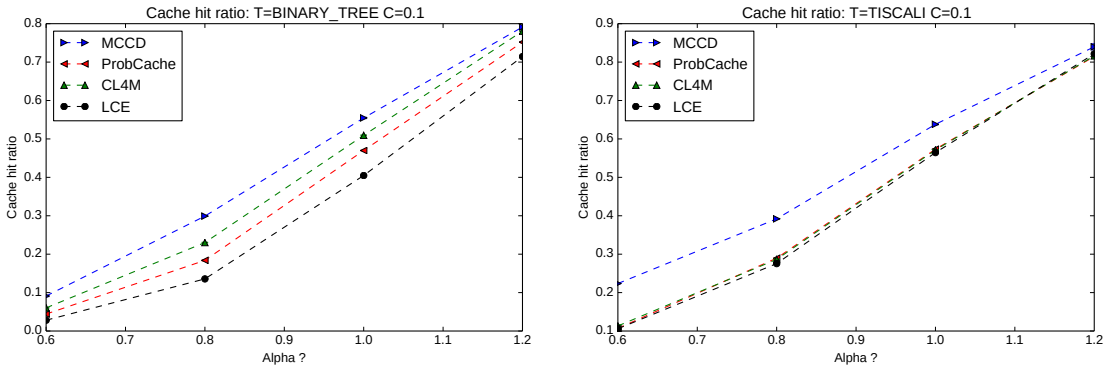


(a) Árvore Binária **(b)** TISCALI

Figura 5. Taxa de Acertos - Tamanho do Cache = 1%

No segundo teste, aumenta-se o tamanho do cache dos roteadores de conteúdo para 1%. Como observa-se na Figura 5(a) na topologia de Árvore Binária, a MCCD obteve uma taxa de acertos maior que as demais avaliadas, atingindo cerca de 62% de acertos para o maior valor de α. Com valores de $\alpha = 0.6 - 1.0$, a CL4M manteve-se acima da PROBCACHE, mas com valores acima deste intervalo as duas se mantiveram praticamente com o mesmo desempenho. A LCE ficou novamente em último lugar. Com relação à topologia TISCALI na Figura 5(b), a MCCD continuou com o melhor desempenho, tendo uma taxa de acerto 400% melhor para um valor de $\alpha = 0.6$, chegando a 70% de acertos para $\alpha = 1.2$. Neste teste, a PROBCACHE obteve um desempenho levemente superior a CL4M, ficando aquela em segundo e esta em terceiro. LCE obteve o pior desempenho novamente.

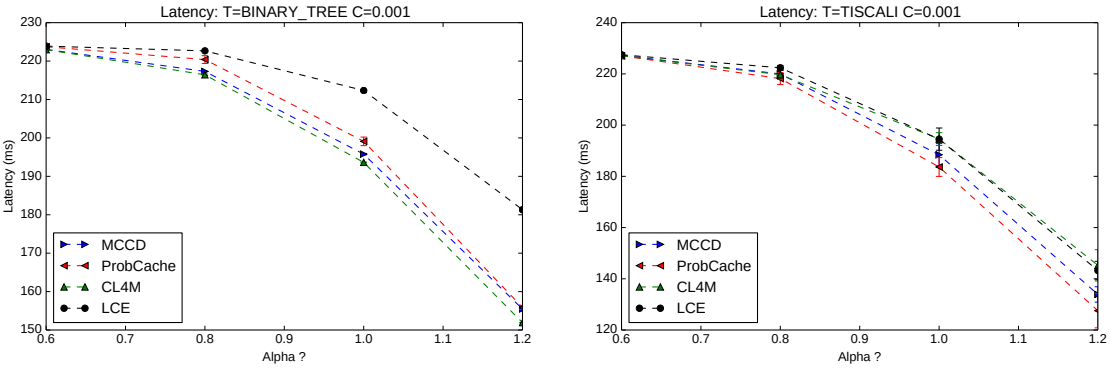
No terceiro teste com um tamanho de cache de 10% na topologia de Árvore Binária, figura 6(a), a MCCD continuou com o melhor desempenho, atingindo cerca de 80% de acertos com $\alpha = 1.2$. A CL4M ficou com o segundo melhor desempenho, seguido



(a) Árvore Binária **(b)** TISCALI

Figura 6. Taxa de Acertos - Tamanho do Cache = 10%

de PROBCACHE e LCE. Neste teste pode-se observar que elas obtiveram resultados bem próximos, mas suas curvas não apresentaram pontos de intercecção. Já na topologia TISCALI conforme a figura 6(b), o desempenho da MCCD foi bastante superior às demais. Com $\alpha = 0.6$, a MCCD foi 209% melhor que a segunda colocada. Considerando todos os valores de α, a MCCD teve uma taxa acima das outras três estratégias, obtendo cerca de 80% de acerto para os conteúdos mais populares. As outras três obtiveram uma taxa de acerto praticamente iguais neste cenário, com uma taxa levemente inferior da LCE para $\alpha = 0.6 - 1.0$.

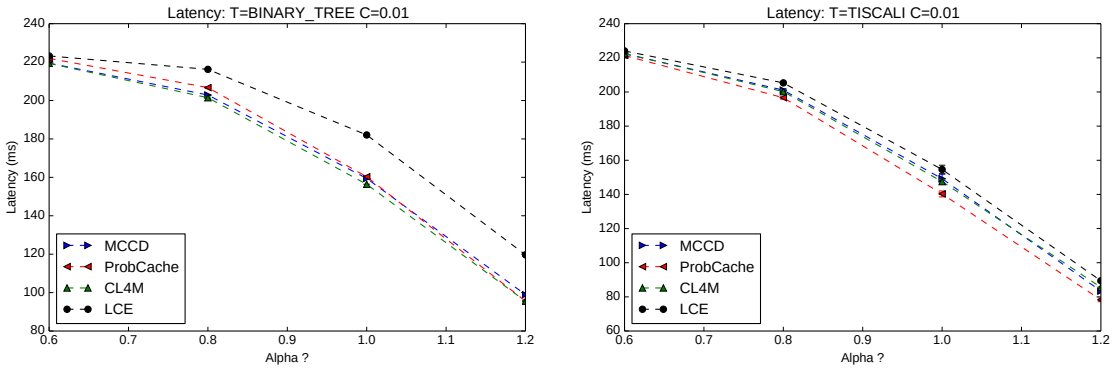


(a) Árvore Binária **(b)** TISCALI

Figura 7. Latência - Tamanho do Cache = 0.1%

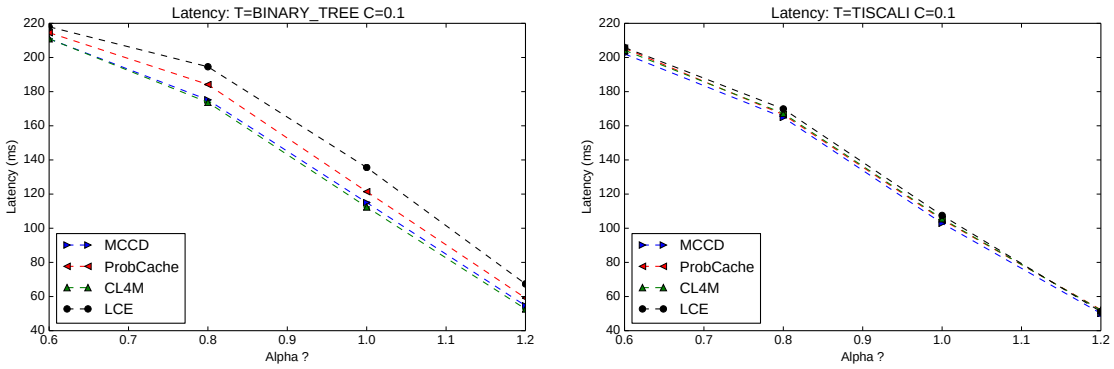
A partir de agora são analisados os resultados para os testes de latência (tempo de *download*), conforme observados na figura 7. Na topologia de Árvore Binária, Figura 7(a), CL4M e MCCD obtiveram um valor de tempo bem aproximados com $\alpha = 0.6 - 0.8$. Para valores maiores, CL4M obteve um tempo menor, sendo que MCCD obteve o segundo melhor tempo, seguido por PROBCACHE e LCE. Na Figura 7(b) para a topologia TISCALI e para $\alpha = 0.6 - 0.8$, PROBCACHE, MCCD e CL4M obtiveram um tempo equivalente. Para valores entre 0.8 - 1.2, PROBCACHE passou a ter o menor tempo, seguido por MCCD, enquanto que CL4M e LCE obtiveram praticamente o mesmo tempo.

A figura 8 mostra a latência para um tamanho de cache de 1%. Em 8(a), percebe-se que os tempos de CL4M e MCCD aproximam-se, tendo agora uma diferença menor quando comparado ao resultado anterior. Nota-se que, em um dado intervalo de $\alpha =$



(a) Árvore Binária **(b)** TISCALI

Figura 8. Latência - Tamanho do Cache = 1%



(a) Árvore Binária **(b)** TISCALI

Figura 9. Latência - Tamanho do Cache = 10%

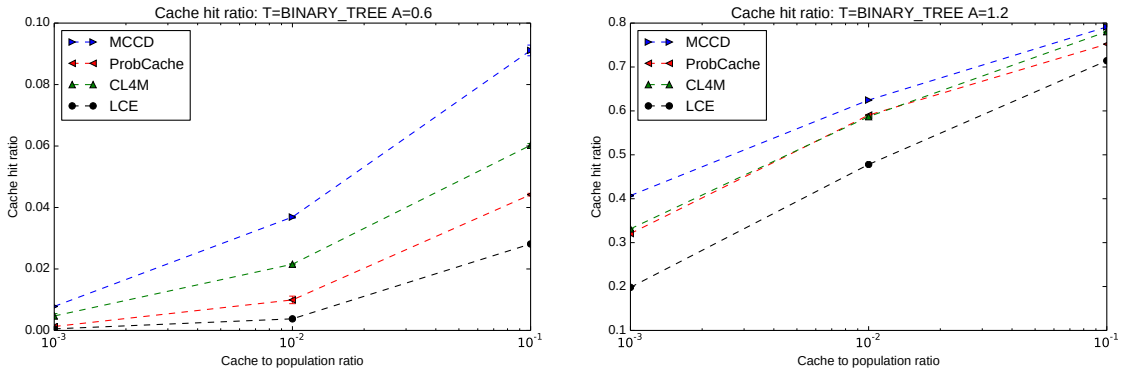
$0.8 - 1.2$, PROBCACHE e MCCD alternam seus resultados, sendo praticamente iguais na maior parte do teste. LCE obteve um tempo bem acima das demais, ficando em último lugar. Já em 8(b), verifica-se que PROBCACHE obteve o menor tempo, ficando MCCD e CL4M com tempos bem aproximados. LCE obteve o maior tempo, mas neste teste ficou bem mais próxima das demais propostas.

Finalmente, avalia-se as topologias com um tamanho de cache de 10%, de acordo com a figura 9. Em 9(a), MCCD e CL4M obtiveram os melhores desempenho, tendo CL4M um tempo um pouco menor quando $\alpha = 0.8 - 1.2$. Diferentemente do teste anterior, PROBCACHE teve um tempo maior, ficando em terceiro lugar e LCE obteve o maior tempo entre todas. Já em 9(b), MCCD obteve o menor tempo entre as propostas para todo α do intervalo considerado, mesmo que a diferença seja bastante sutil. PROBCACHE e CL4M ficaram com o segundo melhor tempo e LCE teve desempenho bem próximo destas últimas, porém, ainda assim, com o maior tempo.

5.1. Discussão dos Resultados

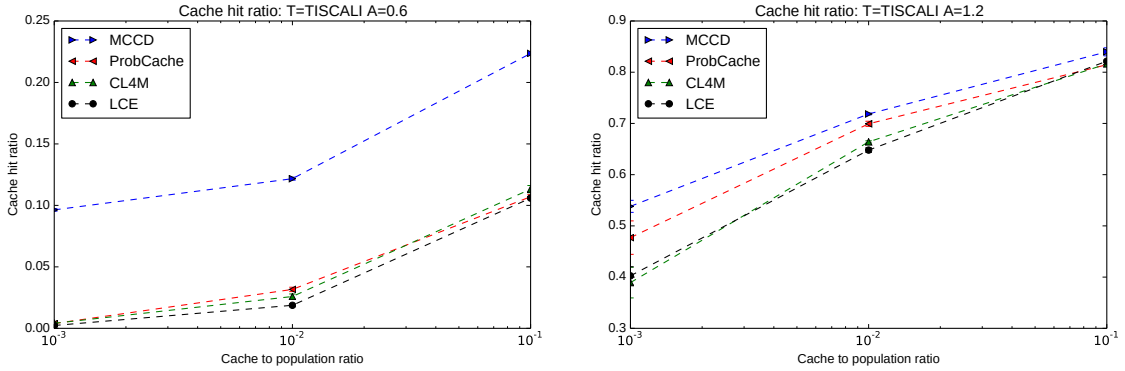
Nesta seção, os resultados são discutidos para as duas métricas avaliadas para as topologias utilizadas. Uma árvore binária é definida por dois parâmetros, os quais são: K, o fator de espalhamento e D, a profundidade da árvores a partir do nó raiz. Em árvores binárias, D impacta diretamente no comprimento dos caminhos entre os nós folhas e a

raiz e K na diversidade de caminhos. Neste tipo de topologia, os nós intermediários à raiz e os nós folhas apresentam praticamente o mesmo valor de centralidade por conta da regularidade apresentada por esta topologia. Desta forma, os nós possuem praticamente a mesma probabilidade para armazenar o conteúdo, sendo que esta probabilidade pode aumentar quando D é pequeno, fazendo com que haja uma diversidade baixa de caminhos para entrega de conteúdo.



(a) Árvore Binária ($\alpha = 0.6$) **(b)** Árvore Binária ($\alpha = 1.2$)

Figura 10. Taxa de Acertos x Tamanho do *Cache*



(a) TISCALI ($\alpha = 0.6$) **(b)** TISCALI ($\alpha = 1.2$)

Figura 11. Taxa de Acertos x Tamanho do cache

Com relação aos resultados obtidos por nossa proposta para este tipo de topologia, observou-se que a MCCD obteve uma taxa de acertos de conteúdo maior devido a possibilidade de armazenar o conteúdo em outros nós alvos que fogem à regularidade da topologia (nós que apresentam praticamente o mesmo valor de centralidade). Desta forma, a MCCD conseguiu obter um resultado de taxa de acertos maior que as demais já que, quando uma métrica não tem um desempenho satisfatório, as outras duas compensam esse baixo desempenho, fazendo com que a MCCD tenha vantagem às demais propostas por causa desse comportamento.

Considerando as figura 10(a), observa-se que a MCCD obteve um valor de taxa de acerto maior para $\alpha = 0.6$, na topologia de Árvore Binária. Isto se deve ao fato de que já que para este valor de α há uma maior quantidade de conteúdos com a mesma popularidade e, portanto, maior número de cópias mantidas em cache na rede. Entretanto,

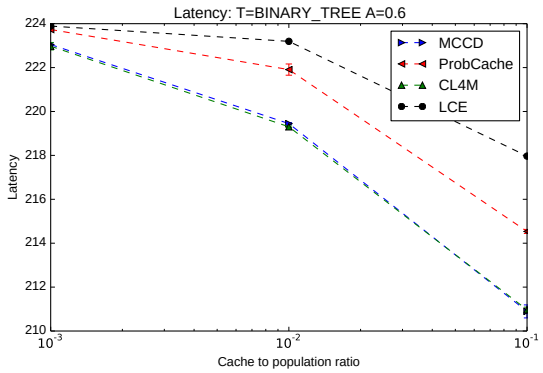
como a MCCD trabalha com mais de uma métrica, isto permite que esse conteúdo possa ser encontrado em outros nós. Ainda, a medida que o tamanho do cache aumenta, este conteúdo tende a ficar por mais tempo armazenado tanto por haver mais espaço disponível quanto também pelo fato da MCCD utilizar como métrica a quantidade de espaço de cache disponível.

Quando consideramos um valor maior de $\alpha = 1.2$, figura 10(b), há poucos conteúdos com a mesma popularidade na rede, por isso que a taxa passa a crescer quase que linearmente. Mesmo assim, a MCCD atinge uma taxa maior do que as outras propostas, dado que os nós que armazenam o conteúdo residem no caminho de requisição de outros nós que também solicitam o mesmo conteúdo, melhorando o desempenho da nossa proposta.

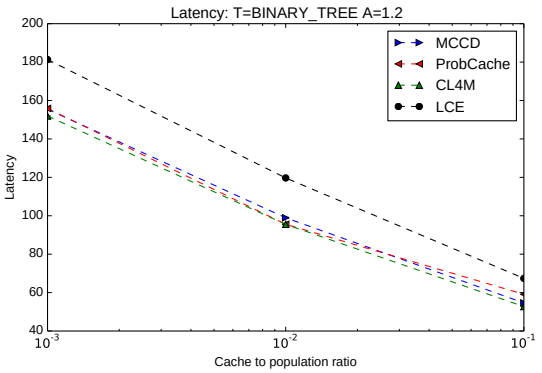
Agora, discute-se os resultados para a topologia TISCALI representado na figura 11. Com o menor valor de $\alpha = 0.6$, figura 11(a), a MCCD obteve uma taxa de acertos crescente à medida que aumenta-se o tamanho do cache dos roteadores. Da mesma forma como na árvore binária, a maior quantidade de conteúdos espalhados pela rede permite que eles sejam encontrados mais facilmente, sendo com uma taxa quase dez vezes maior para o menor valor de cache (0.01%). Isso mostra que a nossa estratégia é bem eficiente quando temos um tamanho bem pequeno alocado para cache. Quando $\alpha = 1.2$ observa-se que ainda assim a MCCD consegue uma taxa de acerto acima das demais para os tamanhos de cache considerados. Por se tratar de um *backbone* intercontinental composto por 161 nós, 328 links e nós com grau médio igual a 4.07 [Domingo-Pascual et al. 2011], essa maior diversidade de características topológicas em relação à árvore binária, a MCCD consegue distribuir o conteúdo mais uniformemente entre os nós que casam com suas métricas pela rede. Além disso, o fato de haver uma menor probabilidade de um nó casar com mais de uma métrica por causa dessa variedade faz com que o conteúdo não se concentre em poucos nós, o que resultaria em uma menor taxa de acertos de conteúdo da rede.

Também, serão discutidos os resultados do tempo de *download* para as duas topologias considerando os mesmos dois valores de α. Na árvore binária, quando $\alpha = 0.6$, figura 12(a), a MCCD obteve um tempo praticamente igual ao do esquema CL4M, que utiliza como métrica a centralidade do nó. Isto ocorre pois, considerando a regularidade da topologia, percebe-se que a métrica centralidade do nó se sobressai às demais e, portanto, as duas propostas obtiveram tempos semelhantes, para os três tamanhos de cache avaliados. Já com um valor de $\alpha = 1.2$, figura 12(b), a MCCD teve um tempo médio ora maior que CL4M e PROBCACHE ora menor que um deles ou ambos. Pode-se atribuir ao fato de que o conteúdo solicitado seja encontrado em nós que casem com as duas outras métricas base da MCCD (tamanho do cache e taxa de acertos) e tais nós podem estar mais distantes do(s) nó(s) que casa(m) com a métrica centralidade do nó.

Para a topologia TISCALI com $\alpha = 0.6$, figura 13(a), a MCCD só teve um tempo de *download* menor quando o tamanho do cache era de 10%. Como os nós apresentam diferentes graus de centralidade, links e tamanho de cache é possível que isso tenha impactado no tempo da proposta. Quando $\alpha = 1.2$, figura 13(b), o tempo a MCCD praticamente permaneceu com o mesmo desempenho de $\alpha = 0.6$. PROBCACHE obteve um tempo menor praticamente que todas as outras propostas, sendo que com o cache de 10% elas convergiram para tempos aproximados.

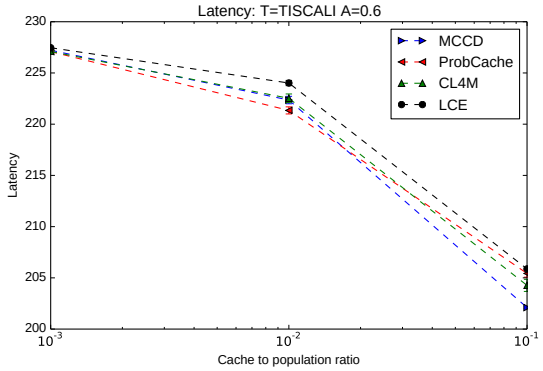


(a) Árvore Binária ($\alpha = 0.6$)

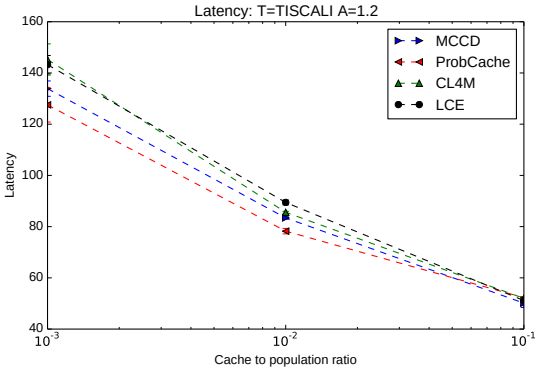


(b) Árvore Binária ($\alpha = 1.2$)

Figura 12. Latência x Tamanho do *Cache*



(a) TISCALI ($\alpha = 0.6$)



(b) TISCALI ($\alpha = 1.2$)

Figura 13. Latência x Tamanho do *Cache*

6. Conclusão e Trabalhos Futuros

A maioria das estratégias de cache em redes ICN consideram apenas em uma métrica para escolher em qual nó armazenar o conteúdo. Dependendo do dinamismo da rede e das requisições de conteúdo, ter apenas uma métrica como base de decisão pode não ser apropriado para atender as requisições dos usuários de uma forma satisfatória. Isso pode trazer uma baixa taxa de acertos de conteúdo, já que os conteúdos são armazenados em poucos nós que casam com tal métrica. Os testes feitos por este trabalho mostraram que as estratégias baseadas em uma métrica não obtiveram uma taxa de acertos maior quando comparadas com nossa proposta, baseada em três métricas.

Os resultados mostraram que tanto para uma topologia genérica quanto para um *backbone* a MCCD obteve uma taxa de acertos maior que as demais e recuperou o conteúdo com um tempo de *download* ora abaixo ora igual às demais estratégias avaliadas. Como trabalhos futuro planejamos otimizar este tempo, bem como o uso de outras métricas que possam melhorar ainda mais nossos resultados como, por exemplo, a popularidade do conteúdo, e a avaliação de outras métricas como, por exemplo, redução do número de salto para a obtenção de conteúdo e redução de solicitações de conteúdos atendidas pelo servidor, mostrando a eficiência da estratégia de cache.

7. Agradecimentos

Agradecemos à CAPES pela ajuda e suporte financeiro dados a este trabalho.

Referências

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. (2012). A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36.

Avelar, E. A. M. and Dias, K. L. Armazenamento em redes orientadas a conteúdo baseado em ranqueamento de caches.

Chai, W. K., He, D., Psaras, I., and Pavlou, G. (2012). Cache "less for more" in information-centric networks. In *NETWORKING 2012*, pages 27–40. Springer.

Cisco (2015). Cisco visual networking index: forecast and methodology, 2014-2019. Technical report, Cisco.

Dannewitz, C. (2009). Netinf: An information-centric design for the future internet. In *Proc. 3rd GI/ITG KuVS Workshop on The Future Internet*.

de Brito, G. M., Velloso, P. B., and Moraes, I. M. (2012). Redes orientadas a conteúdo: Um novo paradigma para a internet. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, 2012:211–264.

Domingo-Pascual, J., Manzoni, P., Palazzo, S., Pont, A., and Scoglio, C. (2011). *NETWORKING 2011: 10th International IFIP TC 6 Networking Conference, Valencia, Spain, May 9-13, 2011, Proceedings*, volume 1. Springer Science & Business Media.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM.

Kutscher, D., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and Waehlisch, M. (2014). Icn research challenges. *Work in progress*.

Lee, S.-W., Kim, D., Ko, Y.-B., Kim, J.-H., and Jang, M.-W. (2013). Cache capacity-aware ccn: Selective caching and cache-aware routing. In *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 2114–2119. IEEE.

Psaras, I., Chai, W. K., and Pavlou, G. (2012). Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60. ACM.

Zhang, G., Li, Y., and Lin, T. (2013). Caching in information centric networking: a survey. *Computer Networks*, 57(16):3128–3141.

Trilha Principal do SBRC 2016
Sessão Técnica 10
Roteamento Interdomínio

Análise da Estabilidade de Ranqueamento de Grau na Rede de Sistemas Autônomos da Internet

Fernando F. Machado[1], Daniel R. Figueiredo[2], Miguel Elias M. Campista[1] *

[1]GTA/PEE-COPPE/DEL-Poli - Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro - RJ - Brasil

[2]LAND/PESC-COPPE - Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro - RJ - Brasil

{ffraga,miguel}@gta.ufrj.br, daniel@land.ufrj.br

Abstract. *Many networks have properties that remain invariant over time, even when many vertices and edges are added and removed. One of such properties is the ranking stability of the most important vertices that can be obtained from network historical data. In this work, we analyze the degree ranking stability over time of the Internet Autonomous Systems network based on techniques from prior works. Additionally, we propose an empirical analysis to evaluate the influence of external factors to the network structure. Results show that the stability of top-ranked vertices is predominantly affected by external factors, while the remaining vertices are also affected by common degree variations in the network. Despite the Autonomous Systems network has been analyzed under several aspects, this work is the first to evaluate the temporal dynamics of the degree ranking.*

Resumo. *Muitas redes possuem propriedades que tendem a se manter ao longo do tempo, mesmo havendo grande variação na quantidade de vértices e arestas. Uma dessas propriedades é a estabilidade do ranqueamento dos principais vértices que pode ser avaliada através dos dados históricos da rede. Neste trabalho, técnicas anteriormente desenvolvidas são utilizadas como base para a análise da estabilidade temporal do ranqueamento de grau da rede de sistemas autônomos da Internet. Adicionalmente, uma análise empírica é proposta para avaliar a influência de fatores externos à estrutura dessa rede. Os resultados mostram que a estabilidade das posições do topo é predominantemente afetada pelos fatores externos, enquanto que as demais posições são também afetadas pelas variações de grau regulares da rede. Apesar da rede de sistemas autônomos ter sido analisada sob muitos aspectos, este trabalho é o primeiro a avaliar a dinâmica temporal do ranqueamento.*

1. Introdução

O estudo empírico de redes reais vem identificando propriedades que se repetem em redes de origens distintas. Uma das propriedades comuns a diversas redes reais, como redes tecnológicas e redes sociais, é a distribuição de grau com cauda pesada, na qual

*Este trabalho foi parcialmente financiado pela CAPES, CNPq, FAPERJ e FINEP.

há a ocorrência de vértices com grau muito superior à média [Figueiredo, 2011]. Dessas redes, as que possuem distribuição de grau próxima a uma lei de potência, ou seja, seguindo o modelo $P(k) \sim k^{-\gamma}$ com $\gamma > 1$ constante, onde $P(k)$ denota a fração de vértices com grau k, são chamadas de redes livre de escala [Barabási e Albert, 1999]. Muitas redes apresentam essa propriedade, tais como a rede formada por *links* de páginas *web* [Albert et al., 1999], a rede de vizinhança de sistemas autônomos da Internet [Siganos et al., 2003] e até mesmo a rede formada por citações de artigos científicos [Redner, 1998].

Um importante aspecto de qualquer rede é o ranqueamento de seus vértices segundo algum critério de importância. Por exemplo, o PageRank é um algoritmo de ranqueamento inicialmente proposto pelo Google para ranquear os vértices da rede de páginas *web* em função da estrutura da rede [Page et al., 1999]. Entretanto redes reais estão constantemente se modificando com a adição e remoção de vértices e arestas. Surge assim o interesse na dinâmica do ranqueamento ao longo do tempo, com a evolução da rede. Em particular, o estudo da estabilidade do ranqueamento, que se refere a propriedade dos vértices manterem suas posições no ranqueamento inalteradas ao longo do tempo.

Uma característica estrutural que se mantém ao longo do tempo em muitas redes é a distribuição de grau de cauda pesada, e em particular na rede de sistemas autônomos que formam a Internet. Mas como essa característica influencia o ranqueamento por grau dos vértices da rede? Ghoshal e Barabási demonstram a existência de vértices com diferenciada estabilidade de ranqueamento, chamados de superestáveis, e ainda que é possível prever a quantidade aproximada desses vértices a partir de um grafo instantâneo da rede [Ghoshal e Barabási, 2011]. Já em outro trabalho, Blumm et al. provam que o nível de estabilidade geral do ranqueamento está relacionado ao ruído do sistema cuja amplitude absoluta pode ser estimada a partir do histórico das variações da característica analisada, que pode ser o grau [Blumm et al., 2012]. Dois aspectos não analisados nesses trabalhos são a variável tempo e o impacto de fatores externos à estrutura da rede, aqui chamados de pontuais. Tanto o tempo quanto os fatores pontuais, que podem afetar um único vértice da rede de maneira relevante, têm influência sobre o ranqueamento da rede mas não foram ainda modelados dada a complexidade.

Este trabalho propõe uma metodologia de análise da estabilidade do ranqueamento de grau considerando o impacto do tempo e dos fatores pontuais em redes de computadores, mais precisamente, na rede de sistemas autônomos da Internet. A metodologia é baseada em três etapas: (i) na análise da existência de vértices superestáveis considerando o tempo, (ii) na análise do impacto do ruído no nível de estabilidade do ranqueamento ao longo do tempo e (iii) na análise dos fatores pontuais que afetam o ranqueamento. Parte da metodologia é baseada nas técnicas dos trabalhos relacionados [Ghoshal e Barabási, 2011, Blumm et al., 2012] e parte é baseada na análise empírica dos dados da rede de sistemas autônomos da Internet. Os resultados confirmaram a presença de vértices superestáveis na rede de sistemas autônomos da Internet, além de demonstrar que os fatores pontuais afetam de forma importante a estabilidade do ranqueamento ao alterar a aptidão dos vértices, em particular quando se trata da rede de sistemas autônomos.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 resume

os trabalhos relacionados. A Seção 3 detalha o conjunto de dados utilizado. A Seção 4 descreve a metodologia empregada na análise. A Seção 5 apresenta os resultados encontrados. A Seção 6 conclui este trabalho e apresenta as direções futuras.

2. Trabalhos Relacionados

O ranqueamento dos vértices de uma rede pode ser realizado em função de diferentes métricas de centralidade, que visam capturar a importância de um vértice na rede. Nessa direção, duas métricas bastante utilizadas são o grau e o PageRank [Boldi e Vigna, 2014]. Apesar do grau inferir a centralidade baseado no número de adjacências de um dado vértice, ele é frequentemente utilizado pela simplicidade e por exigir apenas conhecimento local. O PageRank foi proposto pelo Google com o objetivo de calcular a importância de páginas *web* a partir da estrutura de links na rede *World Wide Web* de forma recursiva [Page et al., 1999]. O valor de PageRank de um vértice é influenciado não apenas pelo seu grau de entrada, mas também pelo valor de PageRank e do grau de saída dos vértices que apontam para ele. Com isso, a importância de um vértice depende também da importância dos vértices que apontam para ele.

A análise da estabilidade de ranqueamento em redes livres de escala foi inicialmente proposta no trabalho de Ghoshal e Barabási [Ghoshal e Barabási, 2011]. Nesse trabalho os autores afirmam que, em redes livre de escala, alguns vértices podem ser considerados superestáveis, ou seja, alguns vértices podem manter suas respectivas posições no ranqueamento por muito tempo durante o qual a rede evolui. A premissa se baseia na capacidade desses vértices manterem as suas posições no ranqueamento de PageRank. Através de simulações numéricas, os autores atestam que, em determinadas redes, as perturbações que alteram a relação de vizinhança entre os vértices sem haver alteração de grau, não alteram a ordem dos vértices no topo do ranqueamento. Isso ocorre quando a diferença entre o PageRank de um vértice superestável e do próximo colocado no ranqueamento é maior do que a variação de PageRank do vértice. A cauda pesada da distribuição de grau pode levar à ocorrência de vértices com grau muito maior do que os demais, que manterão alto PageRank mesmo que as arestas de entrada originem em vértices com baixo PageRank. Isto é, a grande diferença de grau entre os vértices de maior grau garante a estabilidade no ranqueamento desses vértices. Em contrapartida, redes com distribuição de grau exponencial não possuem vértices superestáveis por não apresentarem cauda pesada.

Além de simulação, Ghoshal e Barabási apresentam uma formulação matemática para calcular a quantidade de vértices superestáveis em função da quantidade total de vértices e do expoente da lei de potência que melhor representa sua distribuição de grau. Os resultados calculados através da equação final são aproximadamente iguais aos resultados verificados em simulações numéricas, onde redes sintéticas foram perturbadas de modo a alterar as conexões sem modificar o grau de cada vértice. As quantidades de vértices superestáveis verificadas nas redes livre de escala são pequenas em relação ao total da rede (por exemplo, são menos de dez para redes com 10^7 vértices), mas em geral aumentam com o tamanho da rede. Adicionalmente, uma análise empírica do comportamento do ranqueamento de grau de redes reais ao longo do tempo apontou que a mesma quantidade aproximada de vértices superestáveis manteve seu ranqueamento de grau por um longo tempo enquanto que os demais vértices apresentaram maior instabilidade.

Em outro trabalho, Blumm et al. [Blumm et al., 2012] analisam a dinâmica do

ranqueamento em sistemas complexos, considerando seis diferentes conjuntos de dados de sistemas reais que não representam necessariamente redes. Cada sistema é formado por um conjunto de itens e existe uma característica que é medida individualmente ao longo do tempo para ranquear os itens. Por exemplo, no sistema de citações de artigos é considerada a quantidade de citações recebidas anualmente por cada artigo (item). Repare que esse estudo considera o número de citações que cada artigo recebe em cada ano, sem acumular diferentes anos.

Em um sistema complexo, considere $a_i(t)$ a quantidade absoluta de uma característica de interesse do item i em um instante de tempo t, e $s(t)$ a soma das quantidades absolutas de todos os itens no mesmo instante. Por conseguinte, o valor relativo da característica do item i pode ser calculado como $x_i(t) = a_i(t)/s(t)$, onde $\sum_i x_i(t) = 1$. A partir da análise empírica dos seis conjuntos de dados, Blumm et al. constataram que a dispersão das variações de x a cada passo de tempo é proporcional a x^β, ou seja

$$\sigma_{\Delta x|x} = \sqrt{\frac{1}{R} \times \sum_{\forall i,t|x_i(t)=x}^{R} (x_i(t+1) - x_i(t))^2} \sim x^\beta, \tag{1}$$

onde R é a quantidade de ocorrências de x.

Diante da constatação dada pela Equação 1, os autores formularam uma representação matemática da dinâmica do sistema, onde o valor de x_i é influenciado não só pela aptidão individual do item em relação à qualidade medida mas também pelo ruído presente no sistema. A parcela referente ao ruído do sistema possui amplitude definida pela equação

$$g(x_i) = Bx_i^\beta, \tag{2}$$

onde $g(x_i)$ é a amplitude do ruído que afeta o item i, e os valores do coeficiente B e do expoente β são obtidos a partir de $\sigma_{\Delta x|x}$.

Por fim, os autores concluem que o valor de B está diretamente relacionado à estabilidade do ranqueamento do sistema. Ou seja, quando B é maior do que um limiar a posição dos itens no ranqueamento sofre variações causadas pelo ruído. Dentre os conjuntos de dados avaliados, os sistemas considerados estáveis apresentam valor de B na ordem de 10^{-3} enquanto que os sistemas instáveis possuem B na ordem de 10^{-2} ou 10^{-1}. Já o valor do expoente β foi menor que 1 nos sistemas avaliados, indicando que a amplitude do ruído é proporcionalmente menor nos itens mais próximos do topo do ranqueamento.

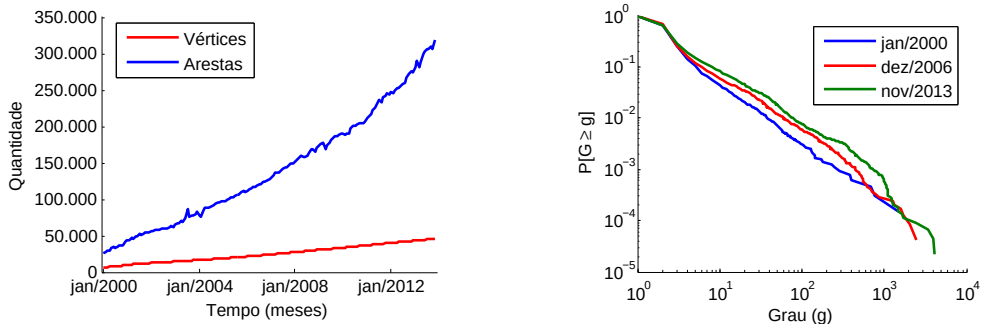
Diferente dos trabalhos anteriores, este trabalho analisa a estabilidade de ranqueamento na rede de sistemas autônomos da Internet levando em conta também o tempo de permanência em cada posição do ranqueamento e o impacto de fatores externos à estrutura da rede.

3. Conjuntos de Dados

Os conjuntos de dados contendo as listas mensais de conexões entre sistemas autônomos da Internet foram obtidos através da página *web* da CAIDA (*The Cooperative Association for Internet Data Analysis*). Os dados disponibilizados pela CAIDA foram inferidos a partir das tabelas de roteamento divulgadas na Internet através do protocolo

BGP (*Border Gateway Protocol*) [CAIDA, 2015]. Neste trabalho, as análises estão baseadas nos dados referentes ao período de janeiro de 2000 a novembro de 2013, por ser o maior período sem interrupção disponível na página da CAIDA. Os dados referentes a novembro de 2014 e novembro de 2015 também são utilizados para uma análise extra.

Neste trabalho a rede de sistemas autônomos é modelada como um grafo, onde os vértices representam os sistemas autônomos e as arestas não-direcionadas são formadas entre sistemas autônomos diretamente conectados. O grafo evolui no tempo, uma vez que o conjunto de dados possui a evolução da rede em fotografias mensais. A partir dessa mesma modelagem, diversos trabalhos observaram que a distribuição de grau da rede de sistemas autônomos se aproxima a uma lei de potência [Chen et al., 2002, Siganos et al., 2003, Ahmad e Guha, 2012]. Entre o início e o fim do período considerado neste trabalho, a rede de sistemas autônomos cresceu mais de 5 vezes em quantidade de vértices e mais de 10 vezes em número de arestas (Figura 1(a)). Apesar disso, a distribuição de grau da rede manteve sua característica livre de escala com pouca variação, conforme ilustrado na Figura 1(b) que apresenta a distribuição de grau da rede no início, meio e fim do período em análise neste trabalho.



(a) Evolução temporal da quantidade de vértices e arestas.

(b) Distribuição de grau da rede.

Figura 1. Caracterização do conjunto de dados de sistemas autônomos da Internet. Apesar do número de vértices e arestas aumentar significativamente ao longo dos anos, a distribuição de grau se mantém com pouca variação.

A Figura 2 apresenta a evolução do ranqueamento dos dez maiores graus da rede a cada mês. Cada linha com cor diferente representa um sistema autônomo diferente. Note que no período em análise, 29 sistemas autônomos diferentes ocuparam as 10 primeiras posições do ranqueamento de grau, demonstrando uma certa instabilidade geral mesmo no topo do ranqueamento. Em particular, repare que o primeiro no ranqueamento em 2000 se manteve na mesma posição por nove anos consecutivos mas ao fim do período ocupou a sexta posição, indicando que fatores pontuais influenciam a aptidão dos sistemas autônomos e consequentemente a habilidade em manter suas posições no ranqueamento. Por outro lado, observa-se que o tempo que os sistemas autônomos permanecem na mesma posição é maior nas primeiras posições do ranqueamento.

4. Metodologia Proposta da Análise de Estabilidade

Os fatores que influenciam a estabilidade do ranqueamento de grau na rede de sistemas autônomos são analisados neste trabalho sob três diferentes aspectos: os vértices

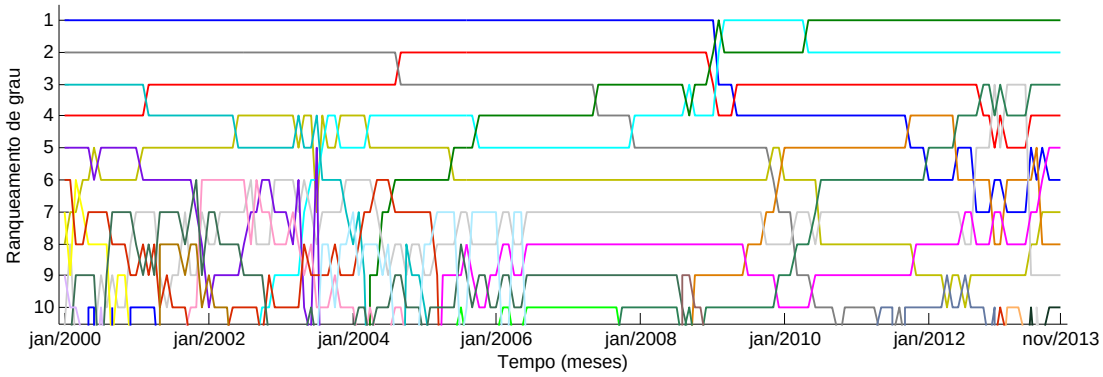


Figura 2. Variações nas dez primeiras posições do ranqueamento de grau na rede de sistemas autônomos da Internet.

superestáveis da rede, o ruído e os fatores pontuais. O objetivo final da análise é compreender como cada característica ou fator afeta o ranqueamento, de modo que seja possível também prever a dinâmica do ranqueamento.

1. **Análise da quantidade e da estabilidade temporal dos vértices superestáveis:** A quantidade prevista de vértices superestáveis (m_c) no período em análise é calculada a partir da equação proposta por Ghoshal e Barabási [Ghoshal e Barabási, 2011]. Os dados de entrada dessa equação são a quantidade total de vértices (N) e o expoente da lei de potência aproximada da distribuição de grau (γ) de uma fotografia da rede. Como essas características da rede podem sofrer alteração ao longo do tempo, neste trabalho são consideradas diferentes fotografias correspondentes a diferentes instantes de tempo. Para estimar γ a partir da distribuição de grau, foi escolhida a técnica MLE (estimador de máxima verossimilhança). Em seguida, para avaliar a relação da estabilidade com o tempo, foi introduzido neste trabalho o cálculo do tempo médio de estabilidade do ranqueamento, que é a média de tempo em que um mesmo vértice se manteve ininterruptamente em cada posição do ranqueamento. Por fim, o tempo médio de estabilidade dos vértices apontados como superestáveis é comparado com o tempo médio de estabilidade dos demais vértices.
2. **Análise do impacto do ruído no ranqueamento:** Conforme proposto por Blumm et al. [Blumm et al., 2012], o nível de estabilidade geral do ranqueamento está relacionado à amplitude do ruído, que é calculada a partir do histórico de variações de grau da rede ao longo do período em análise. Para reproduzir a técnica, mediu-se as variações de grau relativo ($\Delta x|x$) e em seguida extraiu-se o coeficiente de amplitude do ruído (B) a partir da dispersão suavizada de $\Delta x|x$, conforme proposto em [Blumm et al., 2012]. Os programas desenvolvidos foram avaliados através da análise do conjunto de dados de citações da coleção Physical Review [Society, 2015]. A estratégia de refazer os cálculos para o mesmo conjunto de dados analisado no artigo original permitiu comparar os resultados e validar o programa. Neste trabalho, além de calcular o valor de B para todo o período analisado, o mesmo cálculo é feito para períodos menores e os resultados são comparados com as variações observadas no ranqueamento de grau e nas quantidades de adições e remoções de arestas no mesmo período. O objetivo é avaliar se a

variação de B tem correlação com a variação estrutural observada empiricamente na rede.

3. **Análise do impacto de fatores pontuais no ranqueamento:** Fatores pontuais, que influenciam as variações de grau de um único vértice ou de um grupo de vértices, também afetam a estabilidade do ranqueamento da rede. São alguns exemplos de fatores pontuais investigados nesse trabalho:
   - dificuldades financeiras enfrentadas pelas companhias;
   - estratégias comerciais agressivas;
   - estratégias comerciais restritivas;
   - intervenções de órgãos reguladores para evitar concentração no mercado;
   - limitações tecnológicas, como o esgotamento de endereçamento IPv4 em certas regiões.

   Como as origens dos fatores pontuais são muito diversas e suas ocorrências pouco documentadas, é inviável a análise de todos os fatores pontuais de uma rede do tamanho da Internet. Neste trabalho são levantados os fatores pontuais relacionados aos principais vértices do ranqueamento de grau, isto é, aos vértices que ocuparam as primeiras posições do ranqueamento ao longo do período em análise. Após a identificação dos fatores pontuais, o impacto de cada um no ranqueamento de grau é avaliado de forma empírica, através da análise da evolução histórica do grau desses vértices.

A Figura 3 resume as etapas da metodologia empregada na análise de estabilidade do ranqueamento, destacando a contribuição deste trabalho em cada uma das etapas.

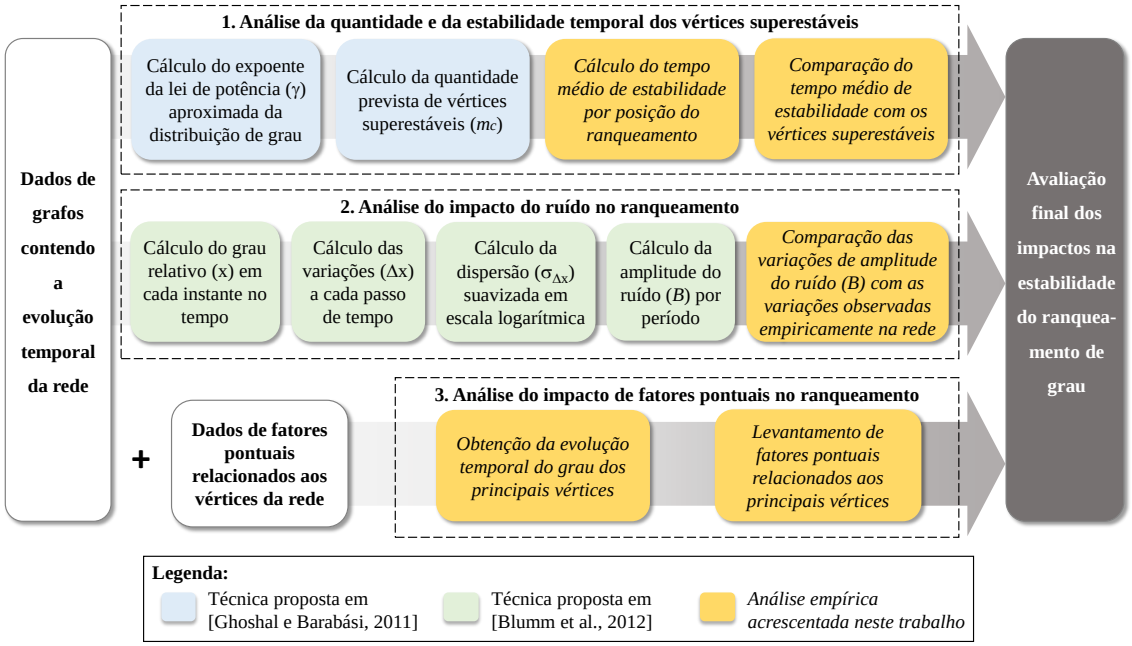


Figura 3. Metodologia de análise da estabilidade de ranqueamento. A metodologia é formada pela análise de três aspectos: os vértices superestáveis da rede, o ruído e os fatores pontuais.

5. Resultados

Nesta seção são apresentados os resultados das análises propostas na Seção 4 para a rede de sistemas autônomos da Internet. Ao final é realizada uma avaliação da previsão feita a partir dos resultados.

5.1. Análise da quantidade e da estabilidade temporal dos vértices superestáveis

A quantidade prevista de vértices superestáveis (m_c) na rede de sistemas autônomos, calculada através da equação proposta por Ghoshal e Barabási [Ghoshal e Barabási, 2011], é de aproximadamente dois vértices em todo o período analisado. Isso quer dizer que as duas primeiras posições no ranqueamento de grau da rede de sistemas autônomos devem apresentar estabilidade muito maior do que as demais posições do ranqueamento. Na Tabela 1 estão os valores de N (quantidade total de vértices) e γ (expoente da lei de potência aproximada da distribuição de grau) utilizados no cálculo de m_c, além do grau médio e dos três maiores graus, referentes ao estado da rede no oitavo mês dos anos de 2000, 2003, 2006, 2009 e 2013. É possível constatar que os três maiores graus são muito maiores do que o grau médio em todos os casos. Em 2009 os três maiores graus estavam muito próximos, sugerindo uma baixa estabilidade no ranqueamento entre eles (fato que pode ser confirmado na Figura 2). Também é observado um aumento da densidade de arestas na rede (grau médio), o que significa que a rede de sistemas autônomos está cada vez mais conectada, aumentando a disponibilidade de sistemas autônomos para trânsito na Internet.

Tabela 1. Propriedades da rede e quantidade prevista de vértices superestáveis em cinco diferentes instantes no tempo.

Rede	Tamanho N da rede	γ aproximado	m_c previsto	Grau médio	Maior grau	2º maior grau	3º maior grau
ago/2000	8.284	2,4	2	4,2	1852	996	814
ago/2003	15.821	2,3	2	4,8	2446	1808	1667
ago/2006	23.102	2,3	2	5,2	2407	2029	1740
ago/2009	32.265	2,3	2	5,7	2487	2350	2241
ago/2013	45.067	2,2	2	6,8	4042	3815	3217

A Figura 4 mostra o tempo médio de estabilidade de cada posição do ranqueamento de grau ao longo de todo o período analisado. As duas primeiras posições do ranqueamento apresentam tempo médio de estabilidade bem superior às demais posições, confirmando o resultado obtido através da equação proposta por Ghoshal e Barabási. Entretanto, ao contrário do que sugere o nome superestável, as duas primeiras posições do ranqueamento não são estáveis por todo o período analisado, pelo contrário, considerando o período total analisado de 167 meses, o tempo médio de estabilidade foi de aproximadamente 42 meses (ou 25% do tempo total) para a primeira posição e 28 meses (ou 17% do tempo total) para a segunda posição. O tempo médio de estabilidade da terceira posição foi de aproximadamente 12 meses (ou 7% do tempo total), sendo menos da metade da estabilidade observada na segunda posição. Apesar da estabilidade das duas primeiras posições do ranqueamento se destacarem, vale ressaltar também que o tempo médio apresenta uma tendência contínua de queda à medida que se observa posições mais distantes do topo do ranqueamento.

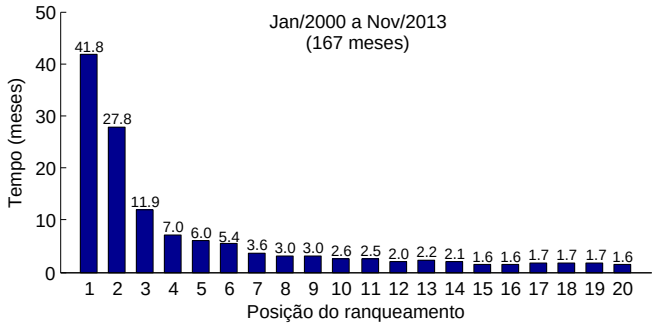
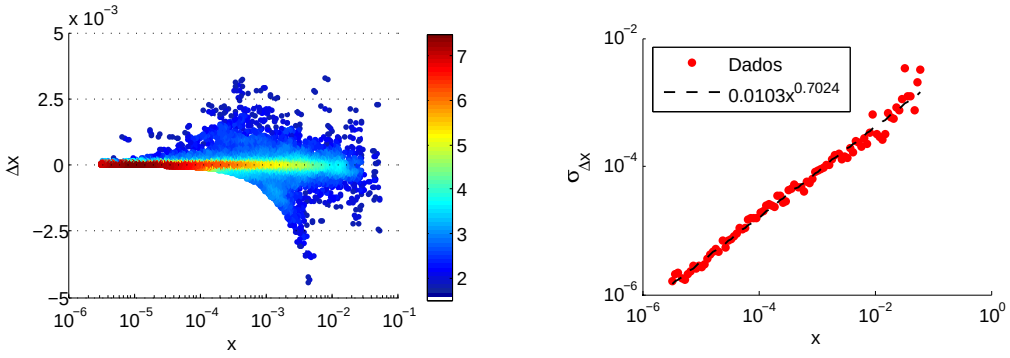


Figura 4. Estabilidade média por posição de ranqueamento na rede de sistemas autônomos.

5.2. Análise do impacto do ruído no ranqueamento

O nível de estabilidade geral do ranqueamento da rede é avaliado através da técnica desenvolvida por Blumm et al. [Blumm et al., 2012]. Para isso, foram calculadas todas as variações entre meses subsequentes dos valores relativos de grau (Δx) e em seguida a dispersão dessas variações ($\sigma_{\Delta x}$). As Figura 5(a) e 5(b) apresentam, respectivamente, os gráficos de Δx e $\sigma_{\Delta x}$ considerando todo o período em análise. Na Figura 5(a) as cores mais quentes indicam maior densidade de pontos. É possível observar uma relativa assimetria em torno de $\Delta x = 0$, onde os pontos relativos aos valores mais altos de x estão mais concentrados abaixo de $\Delta x = 0$. Isso significa que os sistemas autônomos apresentam tendência geral de variar de posição no ranqueamento de grau com o passar do tempo, sugerindo uma baixa estabilidade mesmo no topo do ranqueamento.



(a) Variações do grau relativo (Δx).

(b) Dispersão suavizada das variações de grau relativo ($\sigma_{\Delta x}$).

Figura 5. Características de variação do grau relativo na rede de sistemas autônomos.

A linha tracejada na Figura 5(b) representa a linha de tendência (i.e., regressão linear) dos valores de $\sigma_{\Delta x}$, a partir da qual é estimada a amplitude do ruído (B) presente na rede de sistemas autônomos. Nessa avaliação, considerando todo o período em análise, $B = 0,0103 \sim 1 \times 10^{-2}$. Esse resultado classifica essa rede em uma situação intermediária na classificação de estabilidade proposta por Blumm et al. Isto é, uma parcela das posições próximas ao topo do ranqueamento apresentam estabilidade enquanto que as demais posições apresentam instabilidade. Essa avaliação obtida através do valor estimado de B está coerente com a avaliação do tempo médio de estabilidade por posição do

ranqueamento mostrada na Figura 4, uma vez que as primeiras posições no ranqueamento têm um tempo médio de estabilidade bem superior do que as demais.

Uma questão interessante é a relação entre B e as variações ocorridas na rede. Tal relação foi abordada calculando o valor de B em períodos menores e comparado com as variações detectadas na rede no mesmo período. A Tabela 2 mostra, para cada período de um ano, o valor estimado da amplitude de ruído B, as quantidades proporcionais de arestas adicionadas e removidas na rede, e as quantidades de alterações nas duas primeiras posições do ranqueamento de grau. Para facilitar a visualização, foi acrescentada uma gradação de cor onde o menor valor de cada coluna é marcado de branco e o maior valor de vermelho.

Tabela 2. Amplitude do ruído e variações em cada ano.

Período	Coeficiente B do ruído	% média de arestas adicionadas por mês	% média de arestas removidas por mês	Alterações na 1ª posição do ranquea-mento	Alterações na 2ª posição do ranquea-mento
2000	0,017	11,4%	6,8%		
2001	0,009	9,5%	7,5%		
2002	0,006	7,2%	6,3%		
2003	0,020	8,3%	6,0%		
2004	0,008	6,5%	5,2%		1
2005	0,004	5,4%	4,3%		
2006	0,002	5,1%	3,9%		
2007	0,003	5,2%	3,9%		
2008	0,004	5,9%	4,9%		
2009	0,006	5,3%	4,4%	2	3
2010	0,003	4,6%	4,0%	1	1
2011	0,003	5,0%	3,6%		
2012	0,004	5,2%	4,3%		
2013	0,005	6,0%	4,8%		

A Tabela 2 mostra uma certa correlação entre o valor de B e a quantidade de adições e remoções de arestas. Por exemplo, no ano de 2000, o valor do ruído B é elevado, assim como a porcentagem de adições e remoções. Já em 2006, o valor de B é menor, assim como as porcentagens de adições e remoções de arestas. Por outro lado, não há uma correlação aparente do valor de B com as variações ocorridas nas duas primeiras posições do ranqueamento de grau. Por exemplo, apesar do ano de 2009 ter sido o ano com o maior número de alterações nas primeiras posições, o valor de B não foi dos mais elevados. Isso leva a crer que o ruído tem pouca ou nenhuma influência nas trocas das primeiras posições do ranqueamento dessa rede.

O ruído, da forma que foi definido por Blumm et al., causa variações aleatórias no grau relativo ($x_i(t)$) a cada passo de tempo, que no caso dessa análise é de um mês. A Figura 6 mostra a evolução do grau relativo de todos os sistemas autônomos que ocuparam a primeira e a segunda posições do ranqueamento de grau no período em análise. É possível identificar um ruído de maior amplitude ao longo dos anos de 2000 e de 2003, coerente com os picos do valor estimado de B mostrados na Tabela 2. Note que entre 2000 e 2008 a diferença de grau relativo dos dois primeiros colocados é grande, maior do que os picos de variação causados pelo ruído. A partir de 2009, a diferença entre os dois primeiros colocados se mantém pequena e passam a ocorrer algumas trocas na primeira

posição. Essa condição de proximidade entre o grau da primeira e da segunda posições é inesperada em uma distribuição de grau livre de escala, indicando que a evolução de grau desses vértices sofre influência de fatores pontuais.

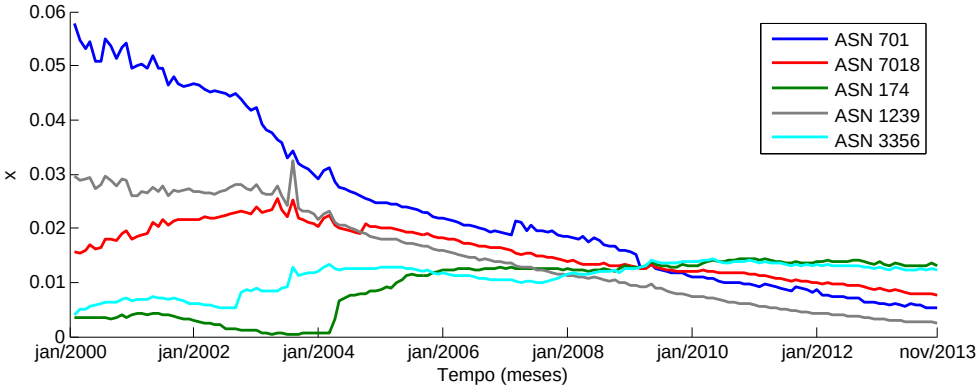


Figura 6. Evolução do grau relativo (x_i) dos principais sistemas autônomos.

5.3. Análise do impacto de fatores pontuais no ranqueamento

Conforme a premissa adotada por Blumm et al. [Blumm et al., 2012], o valor de x_i é determinado não apenas pelo ruído presente no sistema, mas também pela aptidão individual de i. Analogamente, a estabilidade de ranqueamento de cada sistema autônomo depende também da sua capacidade de atrair conexões. Se um sistema autônomo aumentar sua capacidade de atrair conexões em relação aos demais, ele tende a subir de posição no ranqueamento. Se reduzir essa capacidade, tende a perder posição. Apesar de prever a influência da aptidão individual de i no cálculo de x_i, Blumm et al. não definiram uma forma de calcular a aptidão.

Os resultados até aqui indicam que, devido às características da rede, as duas primeiras posições do ranqueamento de grau possuem elevada estabilidade e o nível de ruído do sistema não é determinante para causar as variações observadas nessas posições. Resta, portanto, avaliar se há fatores pontuais responsáveis por alterar a aptidão dos principais sistemas autônomos de modo a influenciar as variações no topo do ranqueamento.

No período em análise, as duas primeiras posições do ranqueamento de grau foram ocupadas pelos sistemas autônomos de ASN (*Autonomous System Number*) 174, 701, 1239, 3356 e 7018. Esses ASNs pertencem a companhias privadas, portanto a expansão das redes desses sistemas autônomos depende fundamentalmente da estratégia de crescimento e da capacidade de investimento de cada companhia. A seguir são listados os fatos relevantes a respeito dessas companhias divulgados publicamente dentro do período analisado:

A) Em 2002 a companhia detentora do ASN 701 (MCI/WorldCom) entrou em concordata [CNN, 2002]. A companhia continuou passando por dificuldades nos anos seguintes, culminando na venda da companhia em 2006 para uma concorrente (Verizon) [Verizon, 2007]. Neste período o grau do ASN 701 encerrou a tendência anterior de crescimento e começou a recuar. Houve uma redução da aptidão desse sistema autônomo.

B) Em 2004 ocorreu a expansão da empresa detentora do ASN 174 (Cogent) para a Europa e a taxa de crescimento do seu grau aumentou significativamente [Cogent, 2015]. Nesse caso ocorreu um aumento da aptidão do sistema autônomo.
C) Entre 2004 e 2005 houve a fusão da companhia detentora do ASN 1239 (Sprint) com uma companhia operadora de telefonia móvel (Nextel). Essa fusão foi considerada problemática, dentre outros motivos, por causa da disparidade das tecnologias das duas redes [Forbes, 2012]. A concentração de esforços na integração das redes de telefonia móvel nos anos seguintes à fusão causou redução de investimentos na rede do sistema autônomo, levando à redução da sua aptidão.
D) Entre 2004 e 2007 a companhia detentora do ASN 3356 (Level 3) adquiriu diversas empresas concorrentes [Wikipedia, 2015]. A partir de 2007 a taxa de crescimento do grau aumentou significativamente como resultado da estratégia de expansão, levando ao aumento da aptidão desse sistema autônomo.

Os órgãos reguladores governamentais têm poder de limitar a atuação das maiores companhias do mercado para estimular a competição, como ocorreu com o sistema Bell na década de 1940. Contudo não foi identificado impacto desse tipo no período em análise. Outro fato relevante é o esgotamento recente do endereçamento IPv4 mas não há impactos evidentes desse tipo nas redes dos cinco principais sistemas autônomos no período em análise.

A Figura 7 mostra a evolução do grau dos principais sistemas autônomos com a indicação da atuação dos fatores pontuais. Nessa figura estão os mesmos sistemas autônomos mostrados na Figura 6, porém com os valores absolutos de grau ao invés dos valores relativos. É possível observar que o fator determinante para a definição das duas primeiras posições do ranqueamento de grau é a taxa de crescimento de grau de cada sistema autônomo, que está associada à aptidão individual do sistema autônomo. Conforme analisado na seção anterior, o ruído apresentou amplitude insuficiente para causar troca nas duas primeiras posições do ranqueamento, com exceção de um pequeno período entre 2009 e 2010 onde o grau dos principais sistemas autônomos esteve muito próximo e o ruído pode ter influenciado momentaneamente o ranqueamento entre eles.

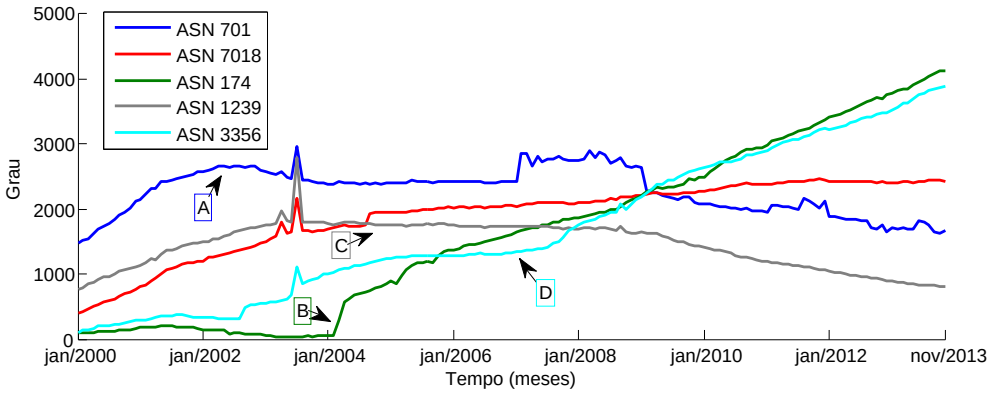


Figura 7. Evolução do grau dos principais sistemas autônomos. As marcações destacam as ocorrências dos fatores pontuais.

5.4. Avaliação da previsão de estabilidade do ranqueamento

Diante dos resultados obtidos, é esperado que as duas primeiras posições do ranqueamento de grau da rede de sistemas autônomos continuem com alta estabilidade em

relação às demais posições. Essa estabilidade será mantida até que ocorram alterações nas aptidões dos principais vértices da rede de forma a causar uma troca nessas posições. E de fato os mesmos sistemas autônomos continuam a ocupar as duas primeiras posições do ranqueamento de grau após dois anos desde o fim do período analisado neste trabalho. A Tabela 3 mostra o ASN e o grau dos cinco sistemas autônomos de maior grau no último mês do período analisado e nos dois anos seguintes, com os dois ASN de maior grau destacados em vermelho. A terceira posição do ranqueamento também foi ocupada pelo mesmo sistema autônomo nos dois anos seguintes, porém a quarta e quinta posições apresentaram variações conforme esperado. Nota-se que o crescimento do grau dos três maiores sistemas autônomos se manteve regular nesse período, indicando que não houve mudança na aptidão desses sistemas autônomos, garantindo a estabilidade do ranqueamento.

Tabela 3. ASN e grau dos 5 sistemas autônomos de maior grau ano final do período analisado e após 1 e 2 anos.

Ranqueamento	Nov/2013		Nov/2014		Nov/2015	
de grau	ASN	Grau	ASN	Grau	ASN	Grau
1º	174	4137	174	4306	174	4765
2º	3356	3897	3356	3990	3356	4284
3º	6939	3408	6939	3598	6939	4270
4º	7018	2433	3549	3573	3549	3560
5º	4323	1710	7018	2340	24482	2694

6. Conclusões e Trabalhos Futuros

A cauda pesada na distribuição de grau da rede de sistemas autônomos é uma característica que se mantêm constante ao longo do tempo. Com isso, o ranqueamento por grau dos vértices leva a grandes diferenças de grau no topo do ranqueamento. Consequentemente, as trocas de posição próximas ao topo do ranqueamento são mais raras do que nas posições mais distantes do topo.

As variações regulares do grau relativo, que podem ser comparadas a um ruído, causam relativa instabilidade no ranqueamento de grau da rede de sistemas autônomos. Porém, como analisado neste trabalho, a estabilidade dos sistemas autônomos nas duas primeiras posições do ranqueamento não é afetada pelo ruído, dado que a sua amplitude é muito menor do que a amplitude da distância média entre os três vértices de maior grau. As variações observadas nas duas primeiras posições foram causadas por fatores pontuais, que alteraram a capacidade individual dos principais sistemas autônomos de atrair conexões, ou em outras palavras, a aptidão. Sendo assim, os fatores aqui chamados de pontuais devem ser levados em conta na análise de estabilidade, principalmente de sistemas autônomos, algo que não foi considerado em trabalhos anteriores.

Como trabalho futuro, conjuntos de dados de outras redes livre de escala serão analisados para confirmar a aplicabilidade global da metodologia utilizada neste trabalho. Em particular, da influência de fatores pontuais na mudança de aptidão dos vértices e consequente instabilidade no ranqueamento.

Referências

Ahmad, M. Z. e Guha, R. (2012). Studying the effects of Internet exchange points on Internet topology. *J Inform Tech Softw Eng*, 2:114.

Albert, R., Jeong, H. e Barabási, A.-L. (1999). Diameter of the world wide web. *Nature Communications*, 2(6749):130–131.

Barabási, A.-L. e Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Blumm, N., Ghoshal, G., Forró, Z., Schich, M., Bianconi, G., Bouchaud, J.-P. e Barabási, A.-L. (2012). Dynamics of ranking processes in complex systems. *Physical Review Letters*, 109(12):128701.

Boldi, P. e Vigna, S. (2014). Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262.

CAIDA (2015). The CAIDA AS relationships dataset. Disponível em `http://data.caida.org/datasets/2013-asrank-data-supplement/`.

Chen, Q., Chang, H., Govindan, R. e Jamin, S. (2002). The origin of power laws in Internet topologies revisited. Em *INFOCOM*, p. 608–617 vol.2.

CNN (2002). WorldCom files largest bankruptcy ever. Acessado 11/2015 em `http://http://money.cnn.com/2002/07/19/news/worldcom_bankruptcy/`.

Cogent (2015). Cogent history. Acessado 11/2015 em `http://www.cogentco.com/en/about-cogent/history`.

Figueiredo, D. R. (2011). Introdução a redes complexas. Em *Atualizações em Informática*, chapter 7, p. 303–358. PUC-Rio.

Forbes (2012). Was Sprint buying Nextel one of the worst acquisitions ever at $35b? Acessado 11/2015 em `http://www.forbes.com/sites/quora/2012/11/29/was-sprint-buying-nextel-one-of-the-worst-acquisitions-ever-at-35b/`.

Ghoshal, G. e Barabási, A.-L. (2011). Ranking stability and super-stable nodes in complex networks. *Nature Communications*, 2(394):1–7.

Page, L., Brin, S., Motwani, R. e Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Relatório Técnico 1999-66, Stanford InfoLab.

Redner, S. (1998). How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B - Condensed Matter and Complex Systems*, 4(2):131–134.

Siganos, G., Faloutsos, M., Faloutsos, P. e Faloutsos, C. (2003). Power laws and the AS-level Internet topology. *IEEE/ACM Transactions on Networking*, 11(4):514–524.

Society, A. P. (2015). APS data sets for research. Disponível em `http://journals.aps.org/datasets`.

Verizon (2007). Verizon Communications 2007 annual report. Acessado 11/2015 em `https://www.verizon.com/investor/app_resources/interactiveannual/2007/note08.html`.

Wikipedia (2015). Level 3 — Wikipedia, the free encyclopedia. Acessado 11/2015 em `https://en.wikipedia.org/wiki/Level_3_Communications`.

Aplicando Redes Definidas por Software à gerência de um ponto de troca de tráfego (IX.br)

Luis Felipe Cunha Martins, Dorgival Guedes

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

{luisf,dorgival}@dcc.ufmg.br

Resumo. *Pontos de troca de tráfego Internet (IX) permitem que diferentes organizações se interliguem para trocar tráfego diretamente em uma certa localidade, visando a reduzir custos com provedores de mais alto nível. Sua operação é uma tarefa complexa e crítica, pelo volume de tráfego presente e pelos interesses comerciais envolvidos. Algumas propostas de utilização do paradigma de Redes Definidas por Software (SDN) no ambiente do IX já foram feitas, mas sem atentar para os desafios operacionais internos dessas instalações. Com base nessa observação, este trabalho tem por objetivo descrever como o princípio de SDN pode ser aplicado internamente a um IX e demonstrar as vantagens desse paradigma do ponto de vista do operador do IX. Tarefas, que em alguns casos podem envolver dezenas de comandos de configuração em diferentes dispositivos, podem ser simplificadas para algumas linhas de configuração de um controlador SDN. A solução proposta foi desenvolvida com base no IX Minas Gerais, do projeto PTTMetro (IX.br), e considerando características dos diversos pontos em operação no país.*

Abstract. *Internet traffic exchange points (IX) allow different organizations to interconnect and exchange traffic directly in a certain location, to reduce costs with upstream providers. Their operation is a complex and critical task, due to the volume of traffic present and the comercial interests in play. Some proposals for the use of Software Defined Networks (SDN) in this context have been made, but no attention has been given to the operational challenges in the IX themselves. Based on that observation, this work describes how the SDN paradigm can be applied in an IX and shows the advantages of that approach from the IX operator's point of view. Some tasks that may require tens of configuration commands in different devices to accomplish can be simplified to a few configuration lines in an SDN controller. The proposed solution was developed based on the operation of the Minas Gerais IX, part of the PTTMetro project (IX.br), considering the characteristics of the various points in operation in Brazil.*

1. Introdução

O grande crescimento e aceitação da Internet possibilitou inúmeros benefícios e inovações, incorporando diversas áreas do conhecimento. No entanto, não eram esperadas essa forte adesão e crescente utilização. Com isso, surgiram diversos problemas relativos à segurança, infraestrutura, topologia e comunicação. Para reduzir esses problemas, os pontos de troca de tráfego (IX) surgiram como parte da infraestrutura da Internet, representando um ponto neutro, central, interconectando redes de diversos Sistemas Autônomos (SA) que a constituem e permitindo uma melhor organização da rede,

redução de custos, maior confiabilidade e segurança para seus usuários. Dessa forma, ao invés de depender do roteamento de pacotes entre diversos provedores intermediários ou da contratação de enlaces dedicados para cada rede a qual um SA deseja se conectar, é possível conectar um SA a todos os outros participantes de um IX através de um único enlace, de capacidade adequada, para o IX.

Na evolução da Internet, os IX firmaram-se como ambientes de produção de suma importância, tendo um papel crítico no ecossistema da Internet. Os IX buscam incentivar a troca de tráfego entre os SA localmente nas cidades, objetivando otimizar o desempenho e a conectividade de seus participantes, mantendo a troca de tráfego o mais localizada possível, evitando que os pacotes percorram grandes distâncias da origem ao destino, diminuindo assim a latência da rede e melhorando a experiência dos usuários. Ao interconectar domínios administrativos diferentes, o IX cria rotas alternativas, anteriormente inexistentes [Akashi et al. 2006], provendo uma conexão redundante para os SA, o que interfere diretamente na conectividade da Internet como um todo [Ceron et al. 2009].

Alguns estudos já abordaram a possibilidade de que Redes Definidas por Software (SDN) poderiam simplificar a operação da rede no roteamento entre domínios. No entanto, dado a escala global da Internet, há certas dificuldades em se aplicar o paradigma de SDN ao roteamento entre domínios, sendo o custo um dos fatores mais limitantes.

Nesse contexto, os IX são ideais para se implementar SDN aplicado às redes de longa distância (WAN). A implantação de SDN, em um único IX, pode gerar benefícios para dezenas a centenas de SA conectados, necessitando apenas trocar a matriz de comutação do IX para afetarmos todos os participantes nele conectados, além de representar um ponto de neutralidade e inovação. Com uma implantação gradual e a capacidade de afetar diversos SA com apenas a troca do switch central, outro benefício direto é o custo significativamente menor para se utilizar a solução.

Em nosso estudo, propomos a construção de uma arquitetura de ponto de troca de tráfego baseado em redes definidas por software, abordando os desafios e apresentando diversos casos de usos passíveis de serem construídos em cima dessa arquitetura para demonstrar os potenciais ganhos com a adoção dessa abordagem. O foco do estudo realizado abrangeu as vantagens providas por essa estrutura para a administração do IX, como simplificação do gerenciamento, menor tempo de resposta, adoção de topologias mais robustas, entre outros, que podem diminuir a demanda na equipe de TI que administra o IX, além de prover um serviço de maior qualidade e mais ágil para os seus participantes. Essas tarefas requerem um grande tempo dos administradores que ficam impossibilitados de dar a vazão necessária às demais requisições ao IX. Ao simplificar e automatizar diversas operações, os operadores do IX conseguirão prover um serviço de maior qualidade aos participantes.

O restante do artigo está dividido da seguinte forma: na seção 2, discutimos os trabalhos relacionados, enquanto a seção 3 apresenta uma breve contextualização dos elementos principais de um IX; em seguida, a seção 4 ilustra como a solução proposta pode simplificar diversas operações do dia-a-dia desse ambiente; O protótipo, implementado nos moldes do IX Minas Gerais, é descrito na seção 5, e a seção 6 discute alguns dos resultados alcançados até o momento; por fim, observações finais são discutidas na seção 7.

2. Trabalhos Relacionados

Alguns estudos na literatura já abordaram se SDN poderia simplificar a operação da rede no roteamento entre domínios, trazendo diversos benefícios, como um controle mais direto sobre o encaminhamento, e solucionando alguns dos problemas relativos ao protocolo BGP. O estudo de [Kotronis et al. 2012] abordou maneiras de como melhorar o roteamento dentro de um único AS, possibilitando uma engenharia de tráfego mais eficiente e o controle remoto do caminho fim-a-fim, utilizando-se um controlador SDN. No entanto, todos esses trabalhos exigem modificar a infraestrutura existente, com a troca de equipamentos, por exemplo.

Outros estudos trataram da utilização de SDN no IX, porém sempre focando o problema no ponto de vista dos participantes, sem considerar o IX como uma entidade no processo. Em [Gupta et al. 2014] abordam o problema de expressão de rotas entre SA que desejam se conectar e como SDN pode ser usado para garantir a segurança e facilitar a expressão dessas rotas. Por sua vez, [Stringer et al. 2013] propõem um roteador distribuído que pode ser usado para implementar a troca de tráfego entre duas ou mais organizações autônomas. [Mambretti et al. 2014] apresentam considerações para a implementação de um IX focando nas questões de chaveamento multiprotocolar entre as organizações.

[Lin et al. 2013] propõem uma solução para a implantação gradual de redes SDN, coexistindo de forma transparente com a rede IP tradicional, lidando com desafios como manter a comunicação entre redes SDN e IP. De forma análoga, [Salsano et al. 2014] discutem a utilização de SDN em backbones IP e a coexistência do encaminhamento em redes SDN e IP tradicional. Os autores argumentam que a rede de um Provedor de Internet necessita de operações mais sofisticadas do que as providas por soluções SDN em camada 2, propondo uma arquitetura capaz de lidar com o roteamento IP e manter a inter-operabilidade entre switches OpenFlow e tradicionais. Assim como esses autores, buscamos estabelecer em nosso trabalho a implantação de SDN no ambiente do IX de forma transparente para os participantes, mantendo a inter-operabilidade entre as redes tradicionais e a rede SDN, de forma que os SA não precisem ajustar a sua infraestrutura às nossas mudanças.

Apesar de compartilharmos do mesmo ideal que os estudos citados — utilizar SDN para inovar e prover novas funcionalidades no IX — o projeto proposto aqui difere dos anteriores, uma vez que eles lidaram com os problemas de infraestrutura, como o roteamento entre múltiplos domínios SDN, ou com um foco restrito nos participantes do IX, apresentando os benefícios, desafios e novas funcionalidades, que podem ser utilizadas pelos SA conectados. No entanto, até onde temos conhecimento, nenhum estudo focou na visão da administração do IX em si, como uma entidade independente, visando levantar os desafios e benefícios da utilização de SDN nas tarefas internas do IX, como a centralização e automatização da configuração dos switches, o gerenciamento dos participantes, englobando desde as ativações de novos participantes até a filtragem de tráfego indesejado e verificação de abusos, a engenharia de tráfego e a adoção de topologias mais robustas do que as geralmente utilizadas em redes de camada 2.

3. Principais elementos de um Ponto de Troca de Tráfego

Usualmente, o IX oferece uma arquitetura simples aos SA conectados. Pode-se visualizar a estrutura interna como sendo uma topologia do tipo estrela, com cada SA se conec-

tando através de um enlace a um conjunto de switches de nível 2 que formam a matriz de comutação [Ceptro.br]. A figura 1 ilustra os diversos elementos essenciais ao funcionamento do IX, discutidos a seguir.

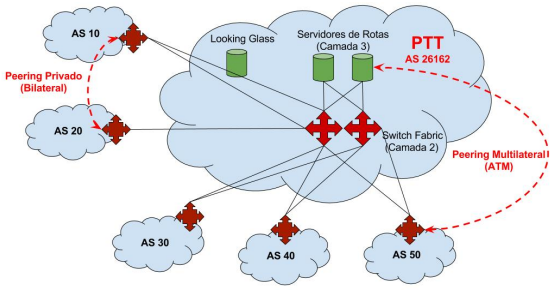


Figura 1. Estrutura interna de um IX, mostrando servidores de rotas, *looking glass* e o escopo de acordos de troca de tráfego (*peering*).

Um único IX é composto por vários PIXes, pontos de interconexão de redes comerciais e acadêmicas, interligados entre si, para formar o IX. Cada PIX representa um ou mais comutadores (switches) em uma posição geográfica, cuja função é prover o meio físico de acesso naquele local.

O anúncio das rotas de cada participante do IX é realizada pelo protocolo BGP através de pontos centrais denominados Servidores de Rotas (SR), de forma que um SA possa realizar a conexão BGP apenas com esses servidores, evitando topologias totalmente conectadas (*full-mesh*) e onerosos custos de interconexão dos roteadores de todos os participantes. A principal função de um SR é comunicar-se com todos os SA e propagar as rotas aprendidas de cada um deles, provendo alcançabilidade entre todos os SA, a partir do estabelecimento de uma única sessão BGP entre o SA e o SR.

Além da troca de tráfego, o IX oferece ainda um serviço conhecido como *looking glass* [LG - Wikipedia], que permite aos administradores dos SA visualizarem todas as rotas divulgadas dentro do IX, sem terem acesso físico à infraestrutura. Caso o participante deseje divulgar suas rotas, ele precisa apenas estabelecer uma sessão BGP com o servidor LG. A grosso modo, o LG nada mais é do que um servidor BGP que aprende rotas mas não repassa o que aprendeu, sendo muito utilizado pelos participantes para solucionar problemas de roteamento e consultar a tabela completa de roteamento do IX.

Uma vez fisicamente conectados no IX, seus membros podem acordar em realizar a troca de tráfego (*peering*) no acordo de troca multilateral (ATM), aberto, com todos os demais membros, ou através de acordos bilaterais (ATB), privado, de natureza seletiva ou restritiva. No primeiro caso, a administração do IX deve garantir a todos os participantes acesso às informações compartilhadas; no segundo, deve garantir uma forma de isolamento do tráfego dentro de sua rede para que o tráfego entre as duas partes flua sem ser acessível aos demais (por exemplo, alocando uma VLAN diferente para cada ATB).

Para entender a importância de IX regionais, pode-se imaginar uma região onde estão presentes alguns SA. Se não há um IX, esses SA estarão trocando o tráfego local, via Internet. Isso pode significar longas distâncias percorridas pelos pacotes. Com a criação de um IX regional, os SA podem conectar-se diretamente, levando à redução da latência, o aumento na resiliência da rede e, provavelmente, economia de custos.

4. Detalhamento da aplicação de SDN à operação de um IX

Com base na experiência diária do IX.br-MG, apresentamos diversos casos de uso construídos para explorar os benefícios do uso de SDN na operação do IX.

Centralizar e automatizar a configuração dos switches: Em um IX tradicional toda a configuração dos switches é feita manualmente. A cada novo participante que deseja conectar-se ao IX, os administradores precisam, entre muitas outras tarefas, habilitar a porta do switch onde ele se ligará, configurá-la para aceitar apenas pacotes oriundos do endereço MAC específico do roteador do participante e adicionar a porta a todas as VLANs necessárias para atender os acordos estabelecidos.

Com o uso de SDN, a adição do membro é enormemente simplificada: após a habilitação da porta, todas as demais configurações seriam adicionadas ao controlador que as aplicaria quando e onde fosse necessário.

Mapear o IX em uma base de dados centralizada: Hoje as informações dos participantes estão cadastradas em um arquivo, sendo boa parte delas utilizadas nas configurações dos equipamentos da rede, como o endereço IP, endereços MAC e se fazem parte do ATM ou não. Além de permitir que haja inconsistências entre as configurações e as informações cadastradas, o sistema atual carece de informações como VLANs criadas, número de VLANs disponíveis e outras.

Na nossa proposta com SDN, todos os dados necessários à configuração da rede estão armazenados em um arquivo, sendo a única fonte de dados para que o controlador configure o plano de dados, evitando inconsistências, além de facilitar a consulta e atualização dos dados. Ao integrar a documentação à configuração da rede, evita-se o armazenamento de dados duplicados e passíveis de erros.

Facilitar a ativação de participantes: Além da configuração do switch, o processo de ativação de um novo participante envolve diversos testes para avaliar a conformidade da conexão física, da conexão BGP e das rotas divulgadas pelo mesmo. Para isolar os testes de ativação, o novo participante é inicialmente colocado em uma área de quarentena na qual existem servidores SR e LG falsos. Apesar de não serem os servidores de produção, esses servidores possuem exatamente as mesmas configurações dos reais. Assim, o participante pode estabelecer as conexões BGP e divulgar as rotas exatamente como o fará quando estiver na produção, e o IX pode avaliar se isso é feito exatamente como esperado, ou se há algum problema. Como no teste de abusos, além de analisar o BGP, outros parâmetros da configuração do participante são avaliados, como o envio de tráfego indesejado e se a conexão do participante suporta aprender os MACs de todos outros participantes. Quando toda a configuração do novo participante é validada, basta migrar sua porta para a VLAN do ATM em que as conexões BGP reais se estabelecerão. Como o processo de ativação é lento, e para evitar que um participante interfira em outro também em ativação, o IX possui diversas áreas de quarentena isoladas entre si. Dessa forma, cada participante é alocado individualmente a uma das áreas durante o processo de ativação.

Por ser uma tarefa complexa, envolvendo analisar o tráfego originado do participante e simular as interações com outros, a ativação pode tirar grande proveito de uma SDN. Esta pode ajudar tanto na geração do tráfego fictício quando na captura dos pacotes enviados pelo participante. Além disso, o isolamento entre os SA durante a ativação se torna trivial, não sendo necessária as migrações do ambiente de quarentena para produção.

Facilitar a detecção e filtragem de tráfego indesejado: Além das configurações de adição de participante mencionadas, são aplicados diversos filtros de pacotes nas portas dos switches para evitar o envio de tráfego indesejado. No IX, o único tráfego oriundo dos participantes aceito são pacotes IP destinados aos demais participantes e as conexões BGP com os SR, além do tráfego ARP. A detecção de outros tipos de tráfego é feita pela análise de dados SFLOW [P. Phaal and Mckee 2001] enviados pelos switches. Em muitos IX, como o de IX.br-MG, o ambiente ainda não está pronto para o SFLOW, sendo a detecção de tráfego indesejado não realizada, deixando o IX mais vulnerável.

Com o uso de uma SDN, todas as regras OpenFlow criadas pela aplicação nos switches realizam o *match* apenas para os tráfegos permitidos em cada tipo de comunicação no IX, rejeitando os demais tipos de tráfego.

Evitar abusos na estrutura: Os participantes necessitam seguir um conjunto de regras para se conectarem à estrutura do IX. Entre elas, não é permitido utilizar a estrutura para a troca de tráfego de um SA consigo mesmo (em duas localidades diferentes, por exemplo), através de duas conexões independentes ao IX e também não é permitido que um SA encaminhe tráfego de redes que ele não anuncia ou direcione a rota *default* para o ambiente do IX.

Em IX menores não há mecanismos implementados para tais checagens, sendo o ambiente vulnerável a tais abusos. Em uma rede SDN, pelo fato de o controlador ter conhecimento de todas as rotas, ele pode rejeitar tráfego de origens cujas rotas não estão anunciadas no IX. Além disso, basta não criar as regras de OpenFlow para a comunicação entre as portas de switches que estejam conectadas a um mesmo participante, impedindo qualquer forma de comunicação entre elementos de um mesmo SA.

Evitar *broadcasts*: O único *broadcast* permitido na rede do IX é do tipo ARP, para a resolução dos endereços MAC na camada 2. Em relação ao ARP, a quantidade de pacotes que trafegam na rede do IX está diretamente relacionada ao número de participantes e de VLANs existentes. Apesar da sua necessidade para o funcionamento da rede, *broadcasts* consomem recursos desnecessários, já que todos os participantes e os respectivos endereços MAC são de conhecimento prévio da administração do IX.

Nesse caso, a SDN impede o trânsito de tráfego *broadcast* não ARP e elimina completamente a propagação dos *broadcasts* ARP. Como o controlador sabe quais participantes estão ativos na rede, ele pode interceptar consultas ARP e gerar as respostas para elas diretamente.

Isolamento do tráfego em acordos bilaterais: São muito comuns os pedidos de VLANs para isolar o tráfego de acordos bilaterais. Em IX com muitos participantes, a quantidade de VLANs começa a atingir os limites dos equipamentos e do próprio cabeçalho dos pacotes. Para eliminar a limitação da quantidade de VLANs, um dos recursos adotados pelos IX é substituí-lo pelo uso de MPLS no encapsulamento do tráfego dos acordos bilaterais, o que implica em custos operacionais e do uso de equipamento especializado.

Na nossa solução com SDN, o isolamento de tráfego é simples de ser alcançado já que o controlador, por ter conhecimento dos acordos bilaterais e o controle de encaminhamento da rede, pode facilmente rotear os pacotes entre os participantes, não sendo necessário o uso do MPLS ou outro recurso avançado.

Engenharia de tráfego entre PIXes: A inexistência de protocolos L2 que aproveitem toda a capacidade da rede em topologias complexas faz com que a maioria dos IX adote uma topologia em estrela, onerosa. Na existência de anéis na rede, utiliza-se do protocolo EAPS [Shah and Yip 2003] para deixar as conexões redundantes em *hot-standby*.

Por conhecer a topologia da rede, as características das conexões e a sua utilização, o controlador SDN permite a adoção topologias mais complexas e baratas, roteando os pacotes entre os PIXes da melhor forma possível e maximizando a utilização das conexões, inclusive utilizando técnicas de balanceamento através dos múltiplos caminhos do anel.

Realizar a tradução e contabilização de VLANs: Os participantes do IX costumam utilizar apenas a quantidade de portas de switch necessárias ao escoamento de todo o seu tráfego, mas sem a saturação das interfaces. Essa prática evita o desperdício de recursos, mas dificulta para a administração do IX contabilizar e tipificar o tráfego em cada porta de switch. O grande problema está na mistura de múltiplos tipos de tráfego em uma mesma porta. Como os participantes podem estar em inúmeros acordos de tráfego diferentes, em uma mesma porta passa tráfego de diversas VLANs, dificultando a contabilização do tráfego de cada VLAN individualmente, pois essa informação demanda recursos que poucas vezes estão disponíveis nos switches.

A solução comumente adotada nos IX, para contabilizar o tráfego de múltiplas VLANs, é separá-las fisicamente utilizando um switch de tradução, onde múltiplas VLANs entram por uma única porta e o tráfego de cada VLAN é encaminhado separadamente para outras portas físicas. Dessa forma, a contabilização pode ser realizada para cada VLAN através da coleta de estatísticas da porta para a qual a VLAN é encaminhada.

Diferentemente da contabilização por VLAN, que depende de implementação específica do fabricante, na rede SDN, este tipo de informação é facilmente obtida. Switches SDN nativos possuem contadores para regras arbitrárias e, para esse caso, bastaria criar regras independentes para cada tipo de fluxo, o que geraria a contabilização apropriada.

5. O protótipo do IX Minas Gerais

Para a validação dos casos de uso e funcionalidades propostas no ambiente SDN, desenvolvemos um sistema para a prototipação rápida e automatizada de topologias semelhantes às adotadas pelos IX no Brasil, mais especificamente o IX.br-MG. Para tanto, optamos por utilizar a ferramenta de criação de redes virtuais Mininet [1], que possibilita instanciar hosts, enlaces e switches.

No protótipo, cada SA foi mapeado em um roteador de borda, capaz de se comunicar via BGP com os SR e LG, sendo representado por um host no Mininet executando o software de roteamento Quagga [2] para divulgar as rotas daquele SA, e, pelo menos, um host para cada rede anunciada para testar a conectividade e o correto anúncio das rotas. Por exemplo, se o ASN1916 anuncia a rota para o bloco 200.131.0.0/24, cria-se um host com o IP 200.131.0.x/24 conectado ao roteador do ASN1916.

Para o protótipo, o ambiente do IX.br-MG foi replicado. Os switches que compõem os quatro PIXes e a rede física do IX são descritos através de switches vir-

[1] `http://mininet.org`
[2] `http://nongnu.org/quagga/`

tuais, conhecidos como Open vSwitch [3]. Cada SR é mapeado da mesma forma que um roteador de borda do SA, ou seja, um host no Mininet, executando o Quagga. O SR reside dentro da rede SDN e sua função é trocar informações, via eBGP, com os roteadores externos e repassá-las à aplicação que gerencia as rotas no controlador. Por sua vez, o LG é instanciado da mesma forma que os SR, alterando-se apenas a configuração do BGP no módulo do Quagga. Os detalhes do mapeamento dos elementos discutidos podem ser visualizados na figura 2. Observe os seis SA, com seus roteadores de borda e "redes internas" conectados ao switches dos cinco PIXes.

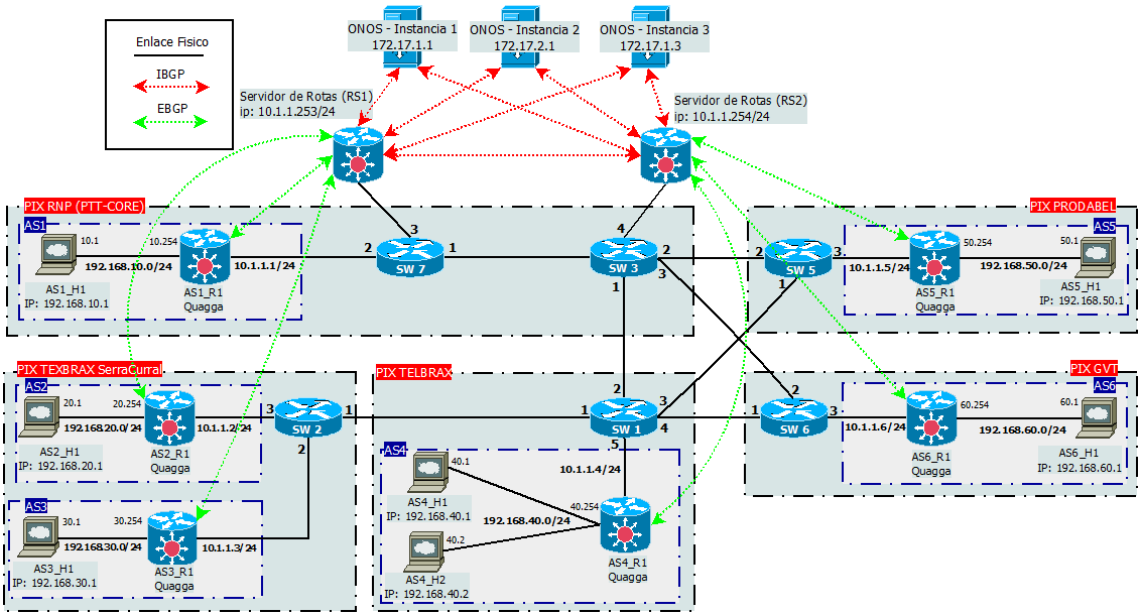


Figura 2. Protótipo do IX.br-MG

O ambiente foi construído pensando na resiliência, alto desempenho e disponibilidade em mente. Dessa forma, após um estudo sobre os controladores existentes, optou-se pelo uso do Open Source Network Operating System (ONOS), um controlador recente, de Dezembro/2014, com foco em operadoras de telecomunicação e redes de longa distância (WAN). Seu desenvolvimento foi norteado por requisitos de alto desempenho, disponibilidade, baixa latência e capacidade de manipulação de grandes redes [Berde et al. 2014].

Os SR atuam como refletores de rotas não se colocando no caminho do tráfego (AS-PATH), provendo a troca de rotas BGP entre os SA e também repassando as rotas ao controlador. Os controladores operam em forma de um cluster, portando-se como uma única entidade para o mundo exterior. Dessa forma, é possível obter tolerância de falhas, tanto no plano de controle, onde um SR pode assumir a demanda do outro em caso de falha, assim como os três controladores entre si, quanto no plano de dados, uma vez que o ONOS monitora os caminhos em utilização, recalculando-os em caso de falhas.

Um das grandes diferenças do ONOS para os demais controladores é a sua abstração chamada ***intents***. O framework de *Intents* é um subsistema que permite as aplicações a especificarem o comportamento da rede, através de diretivas baseadas em políticas - chamadas *intents*. Uma *intent* pode ser definida como **o que** se deseja reali-

[3] `http://openvswitch.org`

zar no lugar de **como** deve ser feito para se alcançar o propósito desejado. Em termos da rede, as *intents* fornecem uma abstração para os desenvolvedores de forma que a aplicação possa requisitar os recursos da rede, sem ter conhecimento de como eles serão executados e disponibilizados, permitindo que os operadores da rede possam 'programar' a rede em alto nível e não através de regras de baixo nível, abstraindo a complexidade da camada de rede. Com o uso de *intents*, acordos de troca de tráfego podem ser representados, usando essa abstração. Devido à sua visão global da rede, o framework de *intents* pode reagir aos eventos da rede. Quando um host deseja comunicar com outro host, cria-se uma *intent* host-to-host e o ONOS calcula o melhor caminho, instalando o fluxo. Na ocorrência de uma falha no caminho entre os dois hosts, o framework recalcula automaticamente um novo caminho e instala a regra correspondente.

Para aproveitar o potencial do framework de *Intents* e integrar o BGP com o ONOS, foi construída uma aplicação, SDX-APP, baseada em um caso de uso já existente no ONOS, para prover conectividade entre os membros, gerenciar a rede do IX e prover o *peering* transparente entre a rede SDN e as demais redes IP tradicionais. Dessa forma, a SDX-APP é responsável por configurar o plano de dados de acordo com as configurações do ambiente do IX, visando à comunicação entre os roteadores de borda dos SA e também deles para os SR e possibilitando a troca de rotas através do BGP.

A aplicação SDX-APP foi construída para integrar a estrutura do protótipo, interagindo com os SR, encaminhando as rotas aprendidas ao ONOS que, por sua vez, transforma-as em *intents*, responsáveis por programar os switches OpenFlow (OF). O objetivo final é prover funcionalidade L3 em uma rede de switches OF, transformando a rede SDN em uma rede de trânsito, capaz de trocar tráfego entre diferentes SA. Um ponto importante é que uma vez que a SDX-APP permite a comunicação transparente entre as redes tradicionais e o IX, através do BGP (e qualquer protocolo legado), os participantes do IX podem aproveitar dos benefícios do SDN em suas redes e realizar a migração de sua infraestrutura para SDN de forma gradual e de acordo com as suas próprias necessidades. A figura 3 descreve a arquitetura da aplicação em alto nível:

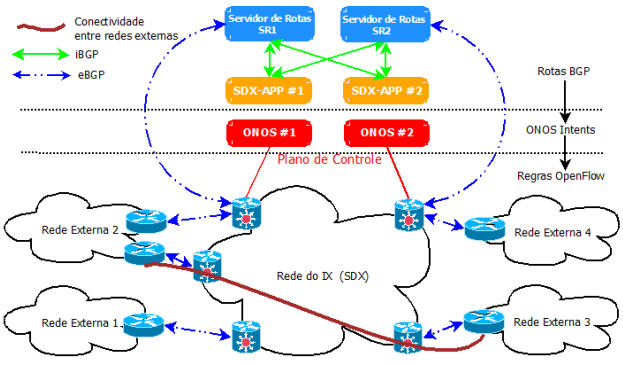


Figura 3. Arquitetura da aplicação SDX-APP

O plano de dados do IX, com as tabelas de fluxo dos switches do núcleo, é mostrado na figura 4. Nela podemos observar as regras para a troca de tráfego entre os participantes e as regras para o encaminhamento do tráfego BGP entre os SA e os SR, assim como a tradução de VLANs quando necessário. Como usamos o framework de *intents* no SDX-APP, é possível aproveitar de seus benefícios, de forma que, sempre que uma *intent*

é solicitada, o controlador checa quais regras já existem e podem ser agregadas com a *intent* solicitada, de forma a diminuir o número de regras no plano de dados.

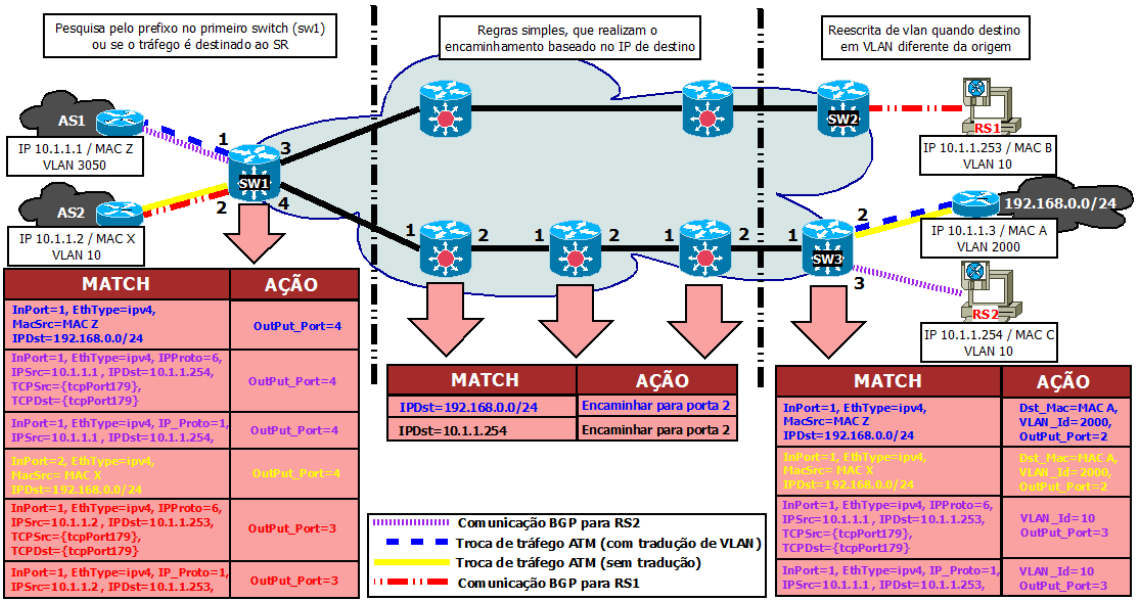


Figura 4. Regras de fluxos instaladas pela aplicação SDX-APP

6. Validação do protótipo

Para a validação do protótipo construído e a sua integração com o BGP e redes IP tradicionais, várias baterias de testes foram realizadas, com os critérios de alta disponibilidade e tolerância a falhas, premissas do ONOS, em mente. Para os testes de alta disponibilidade e tolerância a falhas foram criadas três instâncias do controlador ONOS, implementadas como containers individuais, através do uso da ferramenta de virtualização Docker [4].

6.1. 1ª Bateria - Conectividade Geral

A primeira bateria buscou verificar a conectividade geral entre as redes externas dos participantes, ou seja, se os anúncios de rotas estavam sendo repassados aos SR e se estes repassam as rotas aprendidas aos roteadores de borda de cada participantes, provendo conectividade e a troca de tráfego entre os SA do IX.

Na listagem 1, exibem-se as rotas aprendidas pelas instância do ONOS, mostrando que a SDX-APP recebeu as rotas corretamente, se comunicando via BGP com os dois SR.

```
# Rotas antes da falha do SR1
onos> bgp-routes
    Network            Next-Hop    Origin  LocalPref   MED   BGP-ID          AsPath
    192.168.10.0/24    10.1.1.1    IGP     100         0     10.10.10.31     65006
    192.168.20.0/24    10.1.1.2    IGP     100         0     10.10.10.31     65005
    192.168.30.0/24    10.1.1.3    IGP     100         0     10.10.10.31     65004
    192.168.40.0/24    10.1.1.4    IGP     100         0     10.10.10.31     65003
    192.168.50.0/24    10.1.1.5    IGP     100         0     10.10.10.31     65002
    192.168.60.0/24    10.1.1.6    IGP     100         0     10.10.10.31     65001
```

Listagem 1. SDX-APP BGP-ROUTES - ONOS #1

Quanto ao teste de conectividade, todos os hosts externos foram capazes de comunicarem entre si.

[4] https://www.docker.com

6.2. 2ª Bateria - Falha de um controlador

A SDX-APP, quando executada em um cluster, utiliza os algoritmos de consenso do ONOS para eleger um líder, responsável por gerenciar as rotas BGP e comunicar as informações com as outras instâncias. Dessa forma, no momento da falha, outra instancia deverá assumir o controle, como novo líder, sem prejuízos para o tráfego no IX. Indiretamente, as sessões iBGP entre a instância com problemas e os SR serão removidas.

O funcionamento esperado foi validado pelo teste, conforme mostrado na listagem 2. Toda a conectividade entre as redes externas foi mantida e outro controlador assumiu a liderança de todos os dispositivos, tornando-se líder da SDX-APP. Observe que as sessões estão *up* a vários minutos e que a queda do controlador não interferiu nas sessões iBGP estabelecidas com os demais controladores (172.17.1.1 e 172.17.1.3).

```
# Ping funcionando entre diversos SA diferentes
mininet> AS1h1 ping AS2h1 -c1
PING 192.168.20.1 (192.168.20.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=62 time=80ms

mininet> AS2h1 ping AS4h1 -c1
PING 192.168.40.1 (192.168.40.1) 56(84) bytes of data.
64 bytes from 192.168.40.1: icmp_seq=1 ttl=62 time=111ms

# Sumario BGP no Quagga do SR1
routerServer1> show ip bgp summary
Neighbor      V    AS       MsgRcvd MsgSent TblVer  InQ    QutQ    Up/Down     State/PFxRcd
10.1.1.1      4    65001    56      58      0       0      0       00:02:39    2
10.1.1.2      4    65001    56      58      0       0      0       00:02:38    2
10.1.1.3      4    65001    56      58      0       0      0       00:02:39    2
10.1.1.4      4    65001    56      58      0       0      0       00:02:38    2
10.1.1.5      4    65001    56      58      0       0      0       00:02:39    2
10.1.1.6      4    65001    56      58      0       0      0       00:02:38    2
172.17.1.1    4    65001    56      58      0       0      0       00:02:37    2
172.17.1.2    4    65001    56      58      0       0      0       00:01:07    Active
172.17.1.3    4    65001    56      58      0       0      0       00:02:37    2
```

Listagem 2. Rotas recebidas antes e após a falha do SR1

Após corrigir a falha na instância interrompida, verificou-se que todos os relacionamentos BGP foram restabelecidos com ela.

6.3. 3ª Bateria - Falha de um enlace

Na 3ª bateria de testes, objetivou-se verificar se os controladores são capazes de ajustar o encaminhamento após a falha de um enlace específico. O enlace escolhido foi o que conectava a 3ª porta do switch sw1 do PIX-Telbrax com 1ª porta do switch sw5 do PIX-Prodabel (00000000000000a1/3 ⇔ 00000000000000a5/1), conforme mostrado na figura 2. O comportamento esperado é que a aplicação irá recalcular as rotas para ajustar as *intents* correspondentes, mantendo a conectividade e o encaminhamento do tráfego.

Ao remover o enlace descrito, é possível ver que o controlador averiguou a falha e recalculou os caminhos corretamente, removendo as regras antigas e instalando novas regras de fluxo, de acordo com o novo caminho calculado. Pode-se observar que o caminho antigo (sw1 ⇔ sw5) foi alterado para o próximo caminho mais curto disponível, no caso sw1 ⇔ sw3 ⇔ sw5. O comportamento acima descrito pode ser visto na listagem 3. Conforme a saída do comando *paths*, verifica-se que o caminho foi recalculado pelo ONOS corretamente após a falha do enlace. O caminho entre sw1 e sw5 era composto por um enlace direto, de custo igual a 1 e, após a queda do enlace, o caminho foi modificado para um novo de custo igual a 2, sendo o caminho mais curto disponível.

```
# Caminho utilizado pelo ONOS para comunicar o sw1 com o sw5 antes da falha do enlace
onos> paths of:00000000000000a1 of:00000000000000a5
of:00000000000000a1/3-of:00000000000000a5/1; cost=1.0


# Caminho utilizado pelo ONOS para comunicar o sw1 com o sw5 apos a falha do enlace
onos> paths of:00000000000000a1 of:00000000000000a5
of:00000000000000a1/2-of:00000000000000a3/1⇒00000000000000a3/2-00000000000000a5/2;
cost=2.0
```

Listagem 3. 3ª Bateria de testes - Mudança no caminho apos a queda do enlace

A análise das regras de fluxos e um ping em execução contínua durante o teste mostrou que a conectividade foi mantida entre os hosts, não havendo perdas de pacotes tanto no caminho antigo quanto no novo, validando a mudança no plano de dados.

Após o enlace ter sido restaurado, o ONOS recalculou o melhor caminho e alterou as regras de fluxo para o caminho direto entre sw1 e sw5, de custo igual a 1, que é a rota mais curta, atualizando a tabela de encaminhamento dos switches envolvidos.

6.4. 4ª Bateria - Falha de um Servidor de Rotas

Na última bateria de testes programada, verificou-se o comportamento do sistema após a falha de um dos SR, ocasionando a queda das sessões BGP estabelecidas com ele.

Para esse teste específico, os roteadores dos AS1, AS2 e AS3 foram alterados para estabelecerem sessão BGP apenas com o SR1 e não com todos os SR como é realizado normalmente. Desse modo, novos prefixos divulgados por esses SA não serão repassados para os demais, causando o isolamento de novas redes divulgadas por eles. É esperado que o ONOS invalide as *intents* e entradas de fluxos nas tabelas dos switches referentes às rotas já existentes, uma vez que a SDX-APP não receberá as informações das rotas repassadas pelo SR1, isolando os SA em questão. Os prefixos anunciados apenas ao SR1 também serão retirados pelo BGP da tabela de rotas dos roteadores dos demais participantes.

Como esperado, as rotas correspondentes foram retiradas em todos os roteadores de borda dos participantes e invalidada na aplicação SDX-APP, causando o isolamento dos SA afetados. Nos roteadores de borda desses participantes as sessões do BGP foram finalizadas e apenas a rota para rede local, o núcleo do IX, e para a rede externa do próprio participante continuaram em suas tabelas de rotas.

O comando *bgp-routes* exibe os dados dos anúncios recebidos pela SDX-APP através da sessão BGP estabelecida com os SR. Na listagem 4, executado antes e após a falha do SR1, é possível ver que o controlador retirou as *intents* para as redes 192.168.10.0/24, 192.168.20.0/24 e 192.168.30.0/24, uma vez que elas não estão sendo repassadas pelo SR1 (10.10.10.31) à aplicação SDX-APP e ao ONOS. Dessa forma, as rotas para essas redes, na tabela de rotas dos roteadores de borda dos demais participantes, foram retiradas, interrompendo a conectividade aos SA afetados. Por fim, a aplicação SDX-APP atualizou as rotas das redes 192.168.40.0/24, 192.168.50.0/24 e 192.168.60.0/24 como recebidas pelo segundo servidor de rotas (10.10.10.32).

```
# Rotas antes da falha do SR1
onos> bgp-routes
    Network            Next-Hop    Origin  LocalPref   MED   BGP-ID          AsPath
    192.168.60.0/24    10.1.1.6    IGP     100         0     10.10.10.31     65006
    192.168.50.0/24    10.1.1.5    IGP     100         0     10.10.10.31     65005
    192.168.40.0/24    10.1.1.4    IGP     100         0     10.10.10.31     65004
    192.168.30.0/24    10.1.1.3    IGP     100         0     10.10.10.31     65003
```

```
    192.168.20.0/24     10.1.1.2    IGP     100         0     10.10.10.31     65002
    192.168.10.0/24     10.1.1.1    IGP     100         0     10.10.10.31     65001

# Rotas apos a falha do SR1
onos> bgp-routes
    Network             Next-Hop    Origin  LocalPref   MED   BGP-ID          AsPath
    192.168.60.0/24     10.1.1.6    IGP     100         0     10.10.10.32     65006
    192.168.50.0/24     10.1.1.5    IGP     100         0     10.10.10.32     65005
    192.168.40.0/24     10.1.1.4    IGP     100         0     10.10.10.32     65004
```

Listagem 4. Rotas recebidas antes e após a falha do SR1

Ao restaurarmos a conectividade ao SR1, as sessões BGP com ele foram restabelecidas. As rotas aprendidas foram repassadas à aplicação do SDX-APP que gerou corretamente as *intents* para as redes 192.168.10.0/24, 192.168.20.0/24 e 192.168.30.0/24. O ONOS, por sua vez, reinstalou as regras de OpenFlow - criadas a partir das *intents* - na tabela de fluxos dos switches e os roteadores dos SA aprenderam as rotas para as referidas redes novamente, sendo a conectividade restaurada.

7. Conclusão

O presente trabalho abordou a construção de um ponto de troca de tráfego baseado em SDN, operado nos padrões dos IX brasileiros, criados pelo projeto PTTMetro, do NIC.br. O foco do estudo foi nos benefícios providos por essa estrutura, objetivando simplificar as tarefas de administração do IX, diminuindo a demanda na equipe de TI, tornando sua operação mais simples e eficaz e provendo um serviço de maior qualidade e ágil para os SA conectados, fomentando a entrada de novos participantes e a continuidade do projeto.

Com a experiência na operação do IX de Minas Gerais (IX.br-MG), foram analisadas diversas tarefas da rotina de um IX, levantando-se as demandas mais onerosas, seja em tempo ou complexidade, dificuldades, tarefas passíveis de falhas e outros problemas que poderiam ser simplificados e até automatizados através do uso de uma solução SDN nesse ecossistema. Desse estudo, chegou-se a diversos casos de uso e funcionalidades a serem implementadas, englobando tarefas administrativas, operacionais e de segurança, como a adoção de topologias mais robustas na infraestrutura do IX, simplificação do processo de ativação de novos participantes, gerenciamento dos acordos de troca de tráfego (privados e públicos), tradução de VLANs, verificação de tráfego indesejado, entre outros.

Além dos benefícios diretos como a maior automatização das tarefas, evitar erros devido a configuração erradas ou inconsistência e a maior vazão nas demandas dos participantes à administração do IX, outra grande vantagem da abordagem é não exigir qualquer recurso adicional dos equipamentos de encaminhamento no ambiente físico, desde que sejam compatíveis com o protocolo OpenFlow. Isso permite, inclusive, que recursos avançados como MPLS ou tradução de VLANs sejam implementados em software, reduzindo o custo dos equipamentos e aumentando a flexibilidade do ambiente.

Agradecimentos

Este trabalho foi parcialmente financiado por Fapemig, CAPES, CNPq, e pelos projetos MCT/CNPq-InWeb (573871/2008-6), FAPEMIG-PRONEX-MASWeb (APQ-01400-14), e H2020-EUB-2015 EUBra-BIGSEA (EU GA 690116, MCT/RNP/CETIC/Brazil 0650/04).

Referências

Akashi, O., Fukuda, K., Hirotsu, T., and Sugawara, T. (2006). Policy-based BGP control architecture for autonomous routing management. In *Proceedings of the 2006 SIGCOMM workshop on Internet network management*, pages 77–82. ACM.

Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., and Parulkar, G. (2014). ONOS: Towards an open, distributed SDN OS. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 1–6, New York, NY, USA. ACM.

Ceptro.br. PTTMetro - Pontos de Troca de Trafégo Metropolitanos. Disponível em `http://www.ceptro.br/CEPTRO/MenuCEPTROSPPTTMetro`, visitado em 05/12/2015.

Ceron, J., Lemes, L., Granville, L., Tarouco, L., and Bertholdo, L. (2009). Uma solução para gerenciamento de bgp em pontos de troca de tráfego internet. *XXVII Simpósio Brasileiro de Redes de Computadores (SBRC)*.

Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S. P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E. (2014). SDX: A software defined Internet exchange. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 551–562. ACM.

Kotronis, V., Dimitropoulos, X., and Ager, B. (2012). Outsourcing the routing control logic: Better internet routing based on SDN principles. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 55–60. ACM.

LG - Wikipedia. Looking glass server. Disponível em `https://en.wikipedia.org/wiki/Looking_Glass_server`, visitado em 14/06/2015.

Lin, P., Hart, J., Krishnaswamy, U., Murakami, T., Kobayashi, M., Al-Shabibi, A., Wang, K.-C., and Bi, J. (2013). Seamless interworking of SDN and IP. *SIGCOMM Comput. Commun. Rev.*, 43(4):475–476.

Mambretti, J., Chen, J., and Yeh, F. (2014). Software-defined network exchanges (SDXs): Architecture, services, capabilities, and foundation technologies. In *Teletraffic Congress (ITC), 2014 26th International*, pages 1–6. IEEE.

P. Phaal, S. P. and Mckee, N. (2001). Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks. RFC 3176, RFC Editor. Disponível em `https://tools.ietf.org/html/rfc3176`, visitado em 29/11/2015.

Salsano, S., Ventre, P. L., Prete, L., Siracusano, G., and Gerola, M. (2014). OSHI - open source hybrid IP/SDN networking (and its emulation on Mininet and on distributed SDN testbeds). *arXiv preprint arXiv:1404.4806*.

Shah, S. and Yip, M. (2003). Ethernet automatic protection switching (EAPS). RFC 3619, RFC Editor. Disponível em `https://tools.ietf.org/html/rfc3619`, visitado em 29/11/2015.

Stringer, J. P., Fu, Q., Lorier, C., Nelson, R., and Rothenberg, C. E. (2013). Cardigan: Deploying a distributed routing fabric. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 169–170. ACM.

Delivering Application-Layer Traffic Optimization Services based on Public Routing Data at Internet eXchange Points

Danny Alex Lachos Perez[1], **Samuel Henrique Bucke Brito**[1], **Ramon dos Reis Fontes**[1], **Christian Esteve Rothenberg**[1]

[1] Universidade Estadual de Campinas (UNICAMP)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Information & Networking Technologies Research & Innovation Group (INTRIG)
Av Albert Einstein, 400, Cidade Universitária Zeferino Vaz, Campinas, SP, Brasil

{dlachosp,shbbrito,ramonrf,chesteve}@dca.fee.unicamp.br

Abstract. *Application-Layer Traffic Optimization (ALTO) is a recently standardized protocol that provides abstract network topology and cost maps in addition to endpoint information services that can be consumed by applications in order to become network-aware and take optimized decisions regarding traffic flows. In this paper, we propose a public service based on the ALTO specification using public routing information available at the Brazilian Internet eXchange Points (IXPs). Our ALTO server prototype takes the acronym of AaaS (ALTO-as-a-Service) and is based on over 2.5GB of real BGP data from the 25 Brazilian IX.br public IXPs. We evaluate our proposal in terms of functional behaviour and performance via proof of concept experiments which point to the potential benefits of applications being able to take smart endpoint selection decisions when consuming the developer-friendly ALTO APIs.*

1. Introduction

Internet applications like file sharing, real-time communications and those served from Content Delivery Networks (CDNs) rely on some sort of network topology and cost/performance information to select the best nodes in order to optimize the data transfer. Each application, however, uses different means to build such overlay maps without taking into account underlying network topology considerations or network operator insights. Thus, the application endpoint selection is commonly performed based on partial and inaccurate network views or even randomly in some cases, impacting both the application performance and efficient use of the networking infrastructure [Seedorf and Burger 2009].

Aiming to fill this gap, the Application-Layer Traffic Optimization (ALTO) protocol [Alimi et al. 2014] was designed to allow network operators to provide abstract network information to Internet applications, which can consume this useful information to optimize their connectivity decisions in alignment with the network operators' cost interests and traffic engineering practices. The network information is conveyed in the form of abstract Map Services (Network Map and Cost Map) by an ALTO server (see Fig. 1). A Network Map divides all endpoints (e.g., IPv4/IPv6 addresses or prefixes) in Provider-Defined Identifiers (PIDs) and a Cost Map provides the cost between each pair of PIDs so that it is possible to have a ranking of priority or preference among any pair of endpoints.

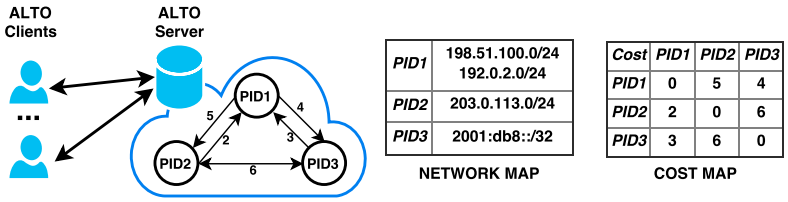


Figure 1. Concept of ALTO and two main services: Network Map and Cost Map.

Most of ALTO implementations today are created by Internet Service Providers (ISPs) based on their individual knowledge of their network dynamics and the costs associated with peering and transit links [Gurbani et al. 2014]. However, recent efforts [Gurbani et al. 2014] show that third parties (not associated with ISPs) can also create valuable Network and Cost Maps from public information.

In this work, we propose a public service called ALTO-as-a-Service (AaaS) that leverages the routing information openly available at Internet eXchange Points (IXPs). Different from related work such as [Madhukar and Williamson 2006] that provides ALTO services through the use of ISP's policies (not available to outside parties), AaaS creates an operator-neutral and public ALTO services based on the BGP routing data from public IXPs operating in Brazil by the IX.br project [Brito et al. 2015]. The raw BGP data is converted into ALTO information, and then stored in a Graph Database (GDB) to be then delivered to ALTO clients through an ALTO server via RESTFul APIs. In order to define the Network Map, each Autonomous System Number (ASN) represents a PID and every prefix (IPv4/IPv6) advertised by an AS correspond to an endpoint. Based on this Network Map, we create different Cost Maps based on the physical or the AS-level topological distance between each pair of ASes.

We evaluate AaaS regarding its functional behaviour (i.e. compliance to the ALTO protocol specifications) and performance profile of our proof of concept prototype implementation based on the OpenDaylight network controller.In addition, we run a series of experiments using a Mininet topology that reflects the BGP-based AS-level connectivity in order to validate the concept and show the potential benefits of applications using the abstract network information from the IXP routing views easily consumable through developers-friendly ALTO APIs.

The remainder of this paper is structured as follows. Section 2 provides an overview of ALTO and includes a primer on IXPs. Section 3 presents the proposed architecture (AaaS) along the basic workflow. Then, the prototype implementation based on BGP data from IX.br is described in Section 4. Section 5 validates the proof of concept with three different experiments (functional, performance, and use case scenario). Section 6 and 7 describe, respectively, current limitations and related work of our research. Finally, we conclude the paper in Section 8 and point to our future work.

2. Background

2.1. ALTO Protocol

Application-Layer Traffic Optimization (ALTO) [Alimi et al. 2014] is a recently Internet Engineering Task Force (IETF) standardized protocol (RFC 7285) with the main goal of exposing network information so that the applications can optimize their endpoint selec-

tions and make informed decisions on questions such as: provided a source IP_{src}, which IP_{dst} endpoints are the best[1] among n candidate destinations.

At a high level, ALTO is an information-publishing interface that fills the gap between networks and applications by allowing network operators to publicly expose abstract network information. This network-to-application information flow benefits both the network providers (i.e. ALTO information providers), who obtain better utilization of their networking infrastructure –provided applications base their endpoint decisions following the ALTO cost maps– and the applications (ALTO information consumers), which do not need to reverse engineer the network and each of them build their own topology maps and endpoint performance rankings.

As shown in the ALTO architecture illustrated in Figure 2(a), an ALTO server gathers network information from multiple sources, such as routing protocols, dynamic and static network information, external interfaces, and so on. These input information is then used to generate an abstract and unified view of the network in the ALTO server, that, in turn, responds to ALTO client requests. The ALTO protocol is based on existing HTTP implementations such as RESTful interfaces for client-server interaction and JSON for request/reply encoding. ALTO Information Services (Fig. 2(b)) include two main services (Map Service: Network Map and Cost Map) and three additional ones:

Network Map: represents a grouping of endpoints into PIDs (Provider-defined Identifier) that may be handled similarly based on its type, topological, physical proximity, or any other criteria. It is responsibility of the ALTO server provider to decide on the grouping of endpoints and the definition and semantics of PIDs.

Cost Map: represents an abstract cost metric (absolute or relative) between any pair of PIDs in the form of path costs. The path cost is a custom-made cost defined and internally computed by the ALTO server implementation.

Map-filtering Service: responsible to filter the result of queries on the Network Map and/or Cost Map to narrow the reply to a subset of PIDs specified by the ALTO client.

Endpoint Property Service (EPS): provides ALTO clients with information about endpoint properties (e.g. which PID belongs a particular endpoint).

Endpoint Cost Service (ECS): unlike the Cost Map (costs between PIDs), it provides information about costs between individual endpoints.

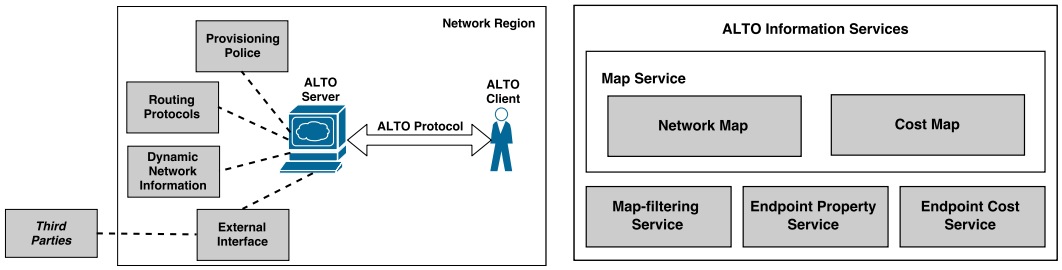
2.2. Internet eXchange Point

An IXP is a shared physical network infrastructure regionally installed with the purpose to facilitate the exchange of Internet traffic between ASes With the traffic exchange as local as possible between different networks that belong to the same region, the number of hops between ASes and dependency on transit providers is reduced.

Brazilian IXPs [Brito et al. 2015] are part of the IX.br project[2], which was created to promote the infrastructure and operational means to increase the connectivity between AS networks of Brazilian metropolitan regions interesting in fostering local Internet traffic exchange. Currently, featuring 25 IXPs in operation, IX.br is the largest IXP ecosystem in

[1]The "best" according to some cost metric defined by the ALTO server provider

[2]http://ix.br



(a) (b)

Figure 2. ALTO (a) Architecture (b) Information Service. Adapted from RFC 7285.

Latin America and it is among the world's top ten both in number of members (1300+)[3] and maximum throughput rate (above 1 Tbps)[4].

The business model —in most cases, including Brazil— adopted by an IXP is open (multilateral peering), commonly allowing anyone to access a large amount of BGP public information through telnet connections to Looking Glass (LG) servers. IP control plane information such as the BGP routing tables, the list of BGP AS-PATH, the community codes, etc., can be retrieved, and by accessing the LGs. This open access to the public routing information is the main input to build the ALTO maps proposed in our work.

3. Design of ALTO-as-a-Service

We propose to deliver ALTO as a public service useful to any application interested in information about AS-level network maps for IP endpoints. The main goal is to generate ALTO PIDs (Network Map) along a ranking of candidate PIDs (Cost Map) using the publicly available BGP routing information at IXPs. To this end, we collect BGP tables (IPv4 and IPv6) and parse the BGP AS-PATH attributes to create the ALTO Network Map and Cost Map respectively. The resulting data structures are stored in a graph-based database (Neo4j) and finally delivered as ALTO services through HTTP Rest APIs.

Figure 3 illustrates the AaaS workflow: (a) Acquiring Input Data, (b) Building Graph Data Models, (c) Creating ALTO Information, and (d) Delivering ALTO Services to serve ALTO client requests. Examples of ALTO clients include, but are not limited to, (1) host running a peer-to-peer file sharing application, (2) tracker in peer-to-peer file-sharing applications, or (3) Software Defined Networking (SDN) controllers.

a) Acquiring Input Data. This first step is to collect BGP routing data publicly available through telnet access to LG servers, using the same methodology proposed in our IX.br ecosystem anatomy [Brito et al. 2015]. The BGP raw data is then pre-processed (formatting, filtering, and assembling) to facilitate its transformation into ALTO data structures.

b) Building Graph Data Models. This process consists of building a suitable connected graph of nodes and relationships to model the ALTO information according to the protocol specification [RFC 7285]. We opt for a property graph[5] since it provides a natural modeling approach to inherent native graph problem at hand. In addition, this approach

[3]http://ix.br/particip, Accessed: September, 2015

[4]http://ix.br/trafego, Accessed: September, 2015

[5]A graph where (i) vertices and edges can have any number of key/value properties, (ii) there can be many types of relationships between vertices and (iii) edges have a directionality.

Table 1. Nodes and its properties in our graph data model.

Name	Properties	Description
PID	Name	Name of a PID
EndPointAddress	Prefix	IP address and the length of the mask
AddressType	Type	"ipv4" or "ipv6"
VersionTag	ResourceID	ID unique for a resource (e.g. a Network Map)
	Tag	Version for a resource

Table 2. Relationships and pairs of nodes that connect them.

Name	Start Node	End Node	Description
Has_PID	VersionTag	PID	Version to which a PID belongs
Has_EndPoint	PID	EndPointAddress	For grouping Endpoints to a PID
Type_EndPoint	AddressType	EndPointAddress	To know the Endpoint type (IPv4/IPv6)
Cost	PID	PID	Path cost between two PIDs

eases the implementation of this model using a Graph DB (Neo4j) as we detail in the following section. Figure 4 provides an overview of the components used in the graph, where nodes are entities that are connected by describing their interactions, i.e. the relationships. Table 1 provides more detailed information about the graph nodes and their properties. Information about the relationships can be found in Table 2. Noteworthy is the relationship labeled as `Cost`, where properties can be included to create different path costs between PIDs resulting in different Cost Maps.

c) Creating ALTO Information. After having the input dataset and the data model ready, the next step is to create the ALTO information and populate the graph DB. In this step, the ALTO server administrator uses the BGP routing information retrieved to create (*i*) grouping of prefixes into PID (Network Map) by ASes, IXPs, BGP communities, points or presence, just to cite a few examples; and (*ii*) defining the preferences / costs between the groups PID (Cost Map) expressed on a path cost such as physical distance between IXPs, topological distances between ASes, etc.

d) Delivering ALTO Services. The last step is deploying an ALTO web server implementing the client-server protocol delivering the REST/JSON APIs to ALTO clients as defined by RFC7285. Internal interfaces to retrieve ALTO information from the GDB are also necessary and we opt to converge on REST/JSON for convenience.

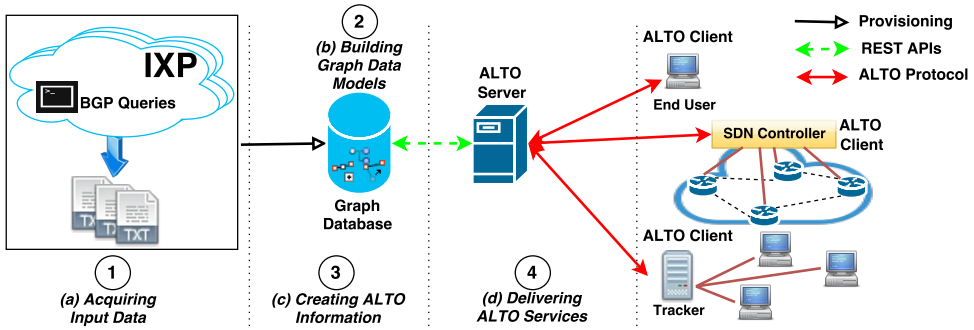


Figure 3. AaaS PoC Workflow

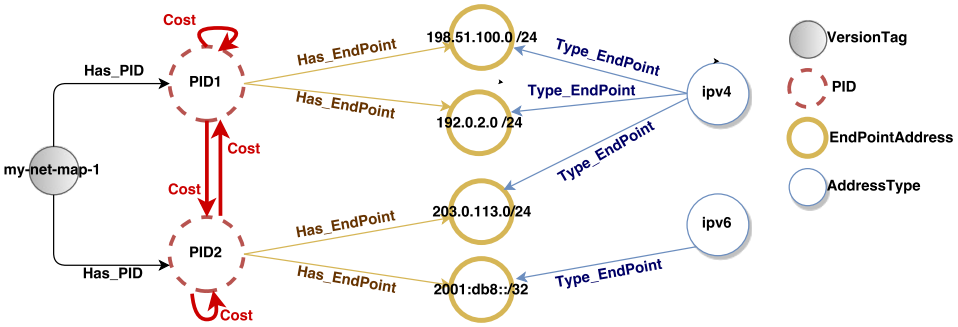


Figure 4. Neo4j Graph Data Model based on RFC 7285 [Alimi et al. 2014]

4. Prototype

We now turn our attention to the implementation choices and prototype details of the AaaS proof of concept. In this section, we provide further details on the BGP data set from the Brazilian IXPs, the Neo4j[6] graph-based database used as the back-end for the ALTO information, and the OpenDaylight[7] (ODL) controller used as ALTO server.

4.1. Input: IX.br BGP Data Set

The data collection methods based on LG remote access to each public IXP in Brazil are those described in [Brito et al. 2015]. More specifically, the dataset (approx. 2.5GB) retrieved during December 2014 was used for our prototype implementation. For data sharing and reproducibility purposes, the raw dataset (including all our 2015 snapshots) and all supporting code are publicly available in our research group repositories.[8]

For the pre-processing job of the raw data, we use a number of Java and R[9] based algorithms. For example, with R we convert the files of IPv4 and IPv6 BGP table into a readable format and exclude prefixes that are advertised by more than 2 ASes.[10] Using Java-based algorithms, we discard the AS-Paths (from the files of the summary list of BGP AS-PATH) which contain the ASN 20121 and ASN 26121 because they are reserved for the IX.br's LG and RS respectively and, therefore, do not participate in Internet routing. Table 3 shows the number of ASes and the number of IPv4 and IPv6 prefixes before and after the pre-processing. As we can see, the amount of discarded data is not significant.

4.2. ALTO Server Backend: Neo4j

As anticipated during the design discussion on the graph modeling approach, using a Graph Database (GDB) is a natural choice to realize the model and embody the ALTO information as a property graph. Furthermore, the ALTO protocol uses a key-value store abstraction for JSON object coding that is very amenable for the Neo4j implementation choice. Neo4j is an open-source non-relational GDB implemented in Java, highly scalable and flexible. It supports true ACID transactions, high availability, and scales to billions

[6]http://neo4j.com/

[7]https://www.opendaylight.org/

[8]https://github.com/intrig-unicamp/ixp-ptt-br/ https://github.com/intrig-unicamp/ALTO-as-a-Service/

[9]http://www.r-project.org

[10]ALTO protocol uses the longest-prefix matching algorithm to compute the mapping from Endpoints to PIDs, therefore a Network Map must not define two or more PIDs that contain an identical Endpoint.

Table 3. Number of ASes and Prefixes (IPv4/IPv6) before and after the dataset pre-processing task at Brazilian IXPs.

	Raw Data	After Proc.	% Out	99% CL ±1 MOE
ASes	49,586	**48,962**	1.26%	12,460
IPv4 Prefixes	563,164	**556,628**	1.16%	16,163
IPv6 Prefixes	21,666	**21,427**	1.10%	9,412

of nodes and relationships [Neo4j 2015]. In addition, application development is highly facilitated through using high-speed traversal query languages such as Cypher.

As shown in Figure 5, to populate the Neo4j GDB, Java-based programs were developed to (*i*) read the input dataset, (*ii*) create the Network and Cost Map, and (*iii*) store the final property graphs into Neo4j using REST interfaces. Next we detail (a) how the endpoints are grouped into PIDs to create the Network Map, and (b) how the path cost between PIDs is computed to create the Cost Map:

a) Network Map. For each pre-processed IPv4 and IPv6 BGP table, the developed algorithm reads each route announcement entry and extracts the ASN (Autonomous System Number) that originated/advertised a prefix. The ASN serves as the PID (location) grouping resulting in a total of 48,962 PIDs (consistent with the current global amount Internet AS). Next, each prefix (be it IPv4 or IPv6) is associated with a particular PID (i.e., AS) considering the origin of the prefix announcement.

b) Cost Map. The path cost between PIDs is calculated as the AS-level topological distance corresponding to the amount of traversing ASes, i.e. path cost equals the number of AS hops between a source and destination AS. A lower cost between PIDs indicates a higher preference for traffic. Two Cost Maps variations are proposed: one which represents the absolute topological distance, and a second one which represents the relative distance. In the latter case, hops between ASes present in the same IXPs are zeroed to favor intra-IXP traffic.

The AS-Path summary files are used to create these maps (see Fig. 5, Cost Map). First, we build the AS-level connectivity graph using an auxiliary GDB. Then, we compute the path cost between each pair of ASes using Cypher queries. Whenever more than one path between two ASes is found, the path with the least number of traversing ASes is chosen. Finally, the path cost is updated in the main ALTO GDB instance.

The Cost Map is a square matrix of order N where N corresponds to the number of PIDs resulting in over 2.3 billion (10^9) relationships labeled as `cost` (two properties are used to distinguish between the absolute and relative distance). Hence, the process of Cost Map creation is partially completed in a proactive manner, while the remainder of the map is created on the fly (or reactively) based on ALTO client requests asking for path cost between specific PIDs.

4.3. ALTO Server Front-End: OpenDaylight

In order to deliver ALTO services, we opt to reuse the OpenDaylight (ODL) controller that features an ALTO project[11] since the Lithium software release. ODL is an open

[11]https://wiki.opendaylight.org/view/ALTO:Main

source SDN controller architecture with production quality code and proven scalable and reliability. The initial release of ALTO in ODL includes, among other modules, ALTO Northbound providing basic ALTO services as RESTful web services (Northbound APIs) for ALTO client/server communications. ALTO Northbound APIs generate ALTO services from data stored in the MD-SAL data store (an ODL core component). For our AaaS implementation, it was necessary to modify the Northbound APIs to generate ALTO services from the data stored in the Neo4j GDB (instead of the MD-SAL topology).

We checkout the *stable/lithium* branch in the ALTO project GitHub repository,[12] which implements the following ALTO services: (*i*) full Map Service, (*ii*) Map-Filtering Service, (*iii*) Endpoint Property Service, and (*iv*) Endpoint Cost Service. To accomplish our proof of concept evaluations, our initial ALTO server delivers the Map-Filtering Service, i.e. the filtered Network Map and the filtered Cost Map, which allow ALTO clients to specify filtering criteria to return only a subset of the full Map Service. Hence, those two Northbound APIs were modified so that the ALTO server retrieves information from the Neo4j backend and converts into the ALTO format specification. All RESTFul API modifications are available in our public repository.

5. Experimental Evaluation

We evaluate our proposal by carrying three different types of experiments using the proof of concept AaaS implementation:[13] (1) functional behaviour (i.e. conformance to ALTO spec.), (2) system performance profiling, (3) emulated IXP use case scenario. Despite using real BGP data, the use case experiments are arguably simplified and mainly serve as a strawman to illustrate the potential of ALTO to deliver useful services to IXP members (and/or third-party applications) to perform better-than-random peer selection.

5.1. Functional Evaluation

We evaluated whether the ALTO server delivers ALTO services in compliance to RFC 7285. For that, we used a REST client tool[14] to retrieve ALTO information in JSON format by communicating with the ALTO server via HTTP request. As discussed in

[12]https://github.com/opendaylight/alto

[13]Single server configuration: Intel® Core™ I7-4790 @ 3.60GHz x 8 with 16GB RAM, running Ubuntu 14.04LTS (Linux) 64-bit.

[14]https://www.getpostman.com/

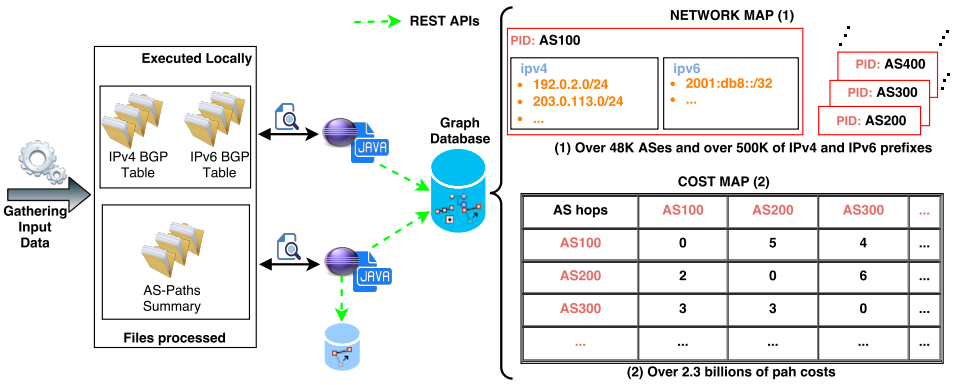


Figure 5. Map Service grouped by AS

the previous section, two RESTful web services are available. An example of the URI, HTTP Method, Content Type, Input Parameters and JSON Response for both the filtered Network Map and the filtered Cost Map (absolute distance) can be seen in Figure 6. To obtain the relative distance, `HopsNumberPTT` instead of `HopsNumber` is used as the input `Cost-metric` parameter.

As expected, our AaaS prototype delivers the ALTO services in accordance with the ALTO specifications and fully reflects the ALTO information stored in Neo4j.

URI : "http://intrig.dca.fee.unicamp.br/controller/nb/v2/alto/filtered/networkmap/my-default-network-map"
HTTP Method : "POST"
Content-Type : "application/alto-networkmapfilter+json"
Input Parameters :

```
{
      "pids" : ["AS100","AS200"]
}
```

HTTP Response :

```
{
   "meta": {
      "vtag": { "resource-id": "my-default-network-map",
        "tag": "da65eca2eb7a10ce" }
   },
   "network-map": {
      "AS100": { "ipv4": [ "192.0.2.0/24","203.0.113.0/24" ],
                 "ipv6": [ "2001:db8::/32" ],
      "AS200": { "ipv4": [ "198.51.100.0/24" ] }
      }
   }
}
```

(a) Filtered Network Map

URI : "http://intrig.dca.fee.unicamp.br/controller/nb/v2/alto/filtered/costmap/my-default-network-map"
HTTP Method : "POST"
Content-Type : "application/alto-costmapfilter+json"
Input Parameters :

```
{   "cost-type" :{"cost-mode": "Numerical",
    "cost-metric": "HopsNumber" },
    "pids" : { "srcs" : [ "AS100" ],
    "dsts" : [ "AS100","AS200","AS300" ] }
}
```

HTTP Response :

```
{
  "meta": {
    "dependent-vtags": [
                { "resource-id": "my-default-network-map",
                          "tag": "da65eca2eb7a10ce" } ],
    "cost-type": { "cost-mode": "Numerical",
                   "cost-metric": "HopsNumber" }},
    "cost-map": { "AS100":
                { "AS100": 0, "AS300": 4, "AS200": 5 } }
}
```

(b) Filtered Cost Map (absolute distance)

Figure 6. HTTP request and JSON response messages to the filtered Network and Cost Map services.

5.2. System Performance Profiling

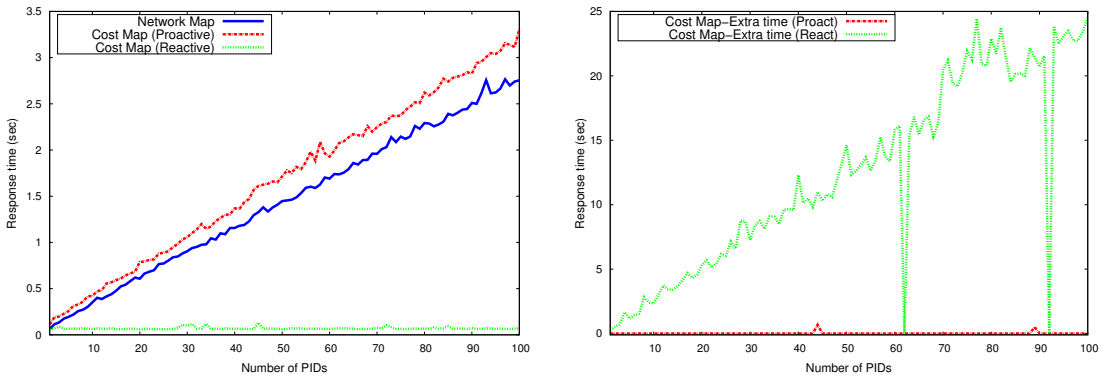
In order to assess the performance of our AaaS prototype, we calculate the response time for the filtered Network and Cost Map (absolute distance) services. For both services, between 1 to 100 PIDs[15] are randomly selected, totaling 100 requests where each request is executed 10 times. The average transaction time is shown in Figure 7.

For the Network Map service, we can observe that the response time increases in proportion to the number of PIDs (See Fig. 7(a), Network Map). For example, when the number of PIDs is 5 and 50, the average time was 0.19 Sec and 1.45 Sec, respectively.

Regarding the Cost Map service, we have two PID input parameters (see Fig. 6(b)). A single PID is used as source PID (`srcs` parameter) and the amount of destination PIDs (`dsts` parameter) varies from 1 to 100. Another relevant factor in the response time is whether a proactive or reactive mode is evaluated.

In order to evaluate the proactive approach, we select a source PID with all possible path costs already created (about 49K path costs). As shown in Figure 7(a), on average, the processing time is 19.73% (or 0.24 Sec) slower compared to the Network Map service. We have a higher query cost, as there are two access operations to the database: one to retrieve the source PID and one to retrieve the path cost with destination

[15]With 50 being the default number of candidate peers in the BitTorrent P2P application.



(a) (b)

Figure 7. (a) Response processing time for the Network and Cost Map (Absolute Distance) services and (b) processing time used to compute two additional steps (the number of hops and insert it into database).

PIDs. It is also necessary to point out that this mode does not have to spend time, i.e. 0 Sec (see Fig. 7(b), Proact) to compute two additional steps (the number of hops and insert them into the database), as they were previously processed.

In the reactive mode, although it also has two access operations, the average processing time is just 0.07 Sec (see Fig. 7(a), React). This behavior can be explained by the fact that the path costs are only created for destinations PIDs of the HTTP request, namely, up to a maximum of 100 path costs. However, for the first request, the time it takes to execute the two additional steps must be considered; for instance, when we send 10 destination PIDs, the processing time is around 2.3 Sec (see Fig. 7(b), React). A particular case is when 62 and 92 destination PIDs are sent, we can see that it takes almost 0 Sec. This is because the selected destination PIDs already had the path cost created with the source PID, so no additional steps were needed. Eventually, when all possible path costs for a particular PID are already created, the response, the time should be in proportional to the number of destination PIDs, as shown in a proactive manner.

Finally, the latency for obtained a response from AaaS can be eased either re-using cached Maps (at the risks of being out-of-date) or running servers in multiple domains through delegation of some ALTO information (e.g. Filtered Network and Cost Maps) to other subdomains.

5.3. Use Case Scenario

We now try to assess the potential effectiveness of AaaS in delivers useful network information be so that an originating peer can make better decisions (in terms of network performance) regarding the candidate destination peers.

Experimental Setup. The network model used is based on a small IXP ecosystem (see Fig. 8) consisting of 22 ASes, each represented by a switch abstraction in the Mininet emulator. A sample AS-Path summary file based on real BGP data was used to create the AS-level connectivity in our experiment topology.[16] The large AS switch represents the

[16]Another noteworthy contribution of our work is the Python-based code developed to generate AS-level topologies for Mininet based on BGP data. The topologies can be used to emulate IXP ecosystems

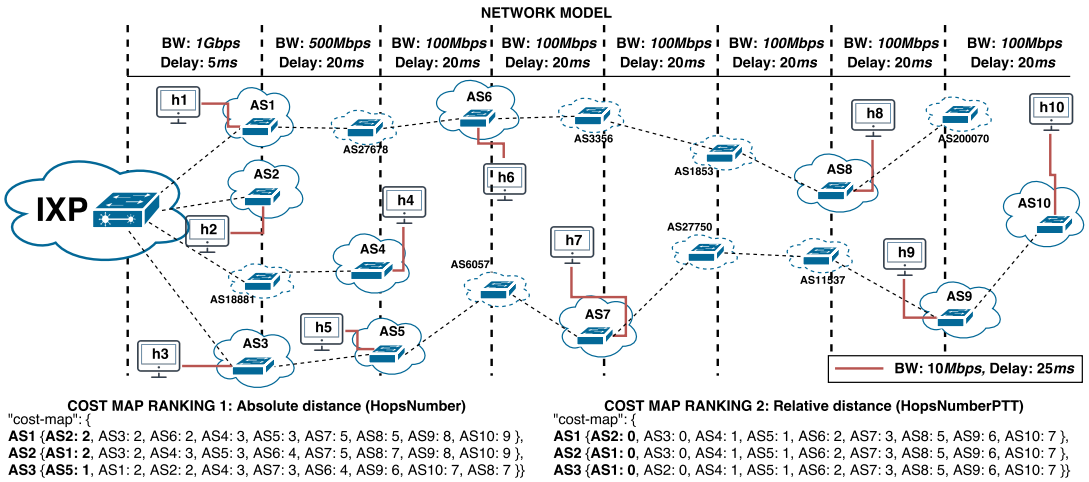


Figure 8. The IXP-based testing network model and the Cost Map rankings based on the absolute and relative distance between ASes.

IXP and then 10 communicating peers are represented as Mininet hosts attached to the (AS abstraction) switches. Links between ASes follow the sample AS-Path attributes and were set with larger bandwidth and lower delay when closer to the IXP.

In the case of ALTO information, the same AS-Path Summary file is used in order to build two Cost Maps based on the topological distance (absolute and relative) expressed as the number of hops between ASes (Fig. 8, Cost Map Rankings). Hosts that belong to ASes present at the IXP (ie., h1, h2, h3) were defined as ALTO clients, since the BGP data (and topology) is mostly meaningful from the IXP vantage point.

Workload and Metrics. For each ALTO client (h1, h2, h3), we run end-to-end round-trip time measurements and available bandwidth with the remaining nine hosts using *ping*[17] and *iperf*[18] tools, respectively. The main idea behind this workload is to emulate a client application (each host) trying to connect with candidate (one of nine possible) peer applications / content servers based on a random selection, and then, compare the obtained bandwidth and latency if the client had use the ALTO information to perform better-than-random peer selection through the use of the ALTO Cost Map ranking.

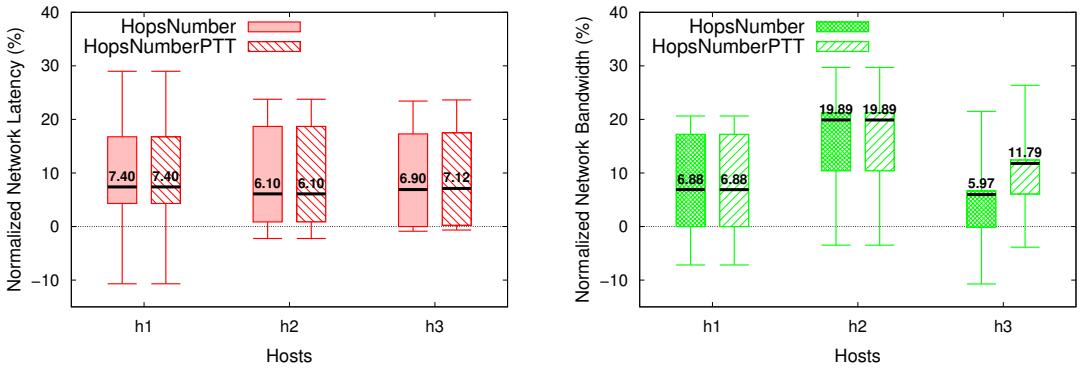
We consider both an ideal scenario without traffic as well as another with a background traffic using the D-ITG traffic generator with randomly selected source and destination pairs send TCP traffic (512 byte packet size, 1,000 pps rate).

Results Analysis. Overall, the results are encouraging as one may expect from applications being able to choose destination peers using ALTO information instead of a random peer selection. Applications with built-in module to evaluate the network performance from/to each candidate peer would correspond to the optimal choice from the application point of view. However, this may not be the best one regarding the network operator policies (e.g., avoid transit costs) and certainly not the simplest (extra code needed per

based on AS-Path Summary files extracted from a LGs. Source code available at: https://github.com/intrig-unicamp/IXP-Brazil-Mininet-Code

[17]Average Round-trip time (ms) of 30 *pings* is computed.

[18]Average TCP throughput (Kbps) in 20 Sec is computed.



(a) Latency using *ping* (b) Throughput using *iperf*

Figure 9. Latency and throughput gains (scenario with background traffic) of h1, h2 and h3 using AaaS Cost Map ranking based on an absolute distance (HopsNumber) and relative distance (HopsNumberPTT) metrics compared to random peer selection.

application) nor the quickest method (the application needs to assess all candidate destination IPs prior to connection setup).

Figure 9 shows the normalized latency and bandwidth (as box plots with mean, quartiles, and max/min values) that an ALTO client would obtain when using the Cost Map ranking compared to a random selection approach (average values of 8 samples is used as baseline) in a scenario with background traffic. Without traffic, results are also positive, more expressive in terms of latency gains (see Table 4) compared to throughput improvements.

Results show an improvement in latency (Fig. 9(a)) and throughput (Fig. 9(b)) of up to 29% could be achieved. In some cases, peers selected through AaaS may end up with slower bandwidth or higher latency (between 1% and 11%). This means that while reducing the total number of AS hops could result in significant performance improvements, shortest paths in the Internet are not always the best (e.g., due to congestion). However, this under-performance represents, on average, less than 25 percent of all cases.

A comparison of performance between two proposed maps (Fig. 8, Cost Map Rankings) is also considered. For example, the Cost Map based on absolute distance (HopsNumber) suggests h3 (AS3) to seek out h5 (AS5), while the Cost Map based on relative distance (HopsNumberPTT) informs h3 (AS3) to connect to h1 (AS1). In both cases, performance improvements above 20% (latency and throughput) are obtained, yet when h3 uses the IXP infrastructure to select a peer (h1), as HopsNumberPTT suggests, further throughput improvements (up to 26%) and lower latency (up to 24%) can be obtained.

6. Related Work

The work in [Khan et al. 2013] shows that it is possible to build the current view of the Internet AS-level topology from the BGP route announcement of AS by using the LG servers. The authors collected raw data from 245 LG servers across 110 countries and build the AS topology. The main lesson learned is that LG-based methods are less error prone than traditional traceroute-based ones, allowing raw data collection and building the AS topology more reliable.

Table 4. Network Latency (ms) in a scenario with no traffic expressed as RTT AVG and ±RTT MDEV.

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10
h1	—	120.5 (±0.4)	120.6 (±0.3)	160.6 (±0.5)	160.6 (±0.5)	180.5 (±0.7)	240.8 (±0.5)	300.7 (±0.8)	360.9 (±0.8)	401.0 (±1.0)
h2	120.5 (±0.3)	—	120.4 (±0.2)	160.6 (±0.4)	160.6 (±0.4)	200.7 (±0.3)	240.9 (±0.6)	320.9 (±0.7)	360.9 (±0.8)	401.1 (±0.6)
h3	120.5 (±0.4)	120.5 (±0.5)	—	160.5 (±0.6)	140.5 (±0.4)	200.6 (±0.5)	220.6 (±0.4)	320.8 (±0.5)	340.8 (±0.4)	380.8 (±0.8)

As shown in Fig. 2(a), the ALTO architecture allows for external interfaces, so that third-parties may feed an ALTO server. The work in [Gurbani et al. 2014] demonstrates how outside parties can create network topology and cost maps for ALTO from public sources of information, specifically using the United States Federal Communications Commission (FCC) public database from the Measuring Broadband America (MBA) program. Similar efforts, e.g., [Pinthong and Lilakiatsakun 2013, Guanxiu et al. 2013] explore the ALTO protocol when mapping IP addresses to an AS to create a network map and BGP route announcements to create a cost.

In the context of P2P, n-Tracker [Shibuya et al. 2011] has been proposed as an ISP-friendly approach based on an ALTO-like server to determine the ranking of candidate peers. Others P2P solutions ([Choffnes and Bustamante 2008, Xie et al. 2007]) provide a reliable network topology information to apply better-than-random peer selection, but not in the form of ALTO services.

7. Conclusions and Future Work

To the best of our knowledge, this is the first work that explores the use of inter-domain routing data publicly available at IXPs to create abstract topology and cost maps following the recently standardized IETF ALTO protocol. Our proof-of-concept implementation is based on the popular Neo4j graph database and the OpenDaylight controller and validated the potential of applications (e.g., P2P clients or trackers, CDNs, SDN controllers) to leverage the network awareness provided by ALTO servers to optimize their decision-making regarding IP endpoint selection resulting better user experience. At the same time, ISPs and ALTO service providers (in our case IX.br operators) can benefit from increased, localized IXP traffic exchange.

Our win-win proposal referred to as ALTO-as-a-Service is not free of limitations and there is a challenging amount of future work. Firstly, the AaaS workflow does not handle periodic updates (e.g. when a new dataset is retrieved) and, our Cost Map rankings are based only on relatively static AS-Path distance and do not consider more dynamic information such as actual bandwidth, latency, packet loss rate, etc. Thus, dynamic updates of cost maps based on public Internet quality measurements (e.g., SIMET Traffic Measurement System, RIPE TTM) are at the top of our research agenda. It is important to stress that while the ALTO protocol may provide dynamic network information, it is not intended to replace near-real-time congestion control protocols [Alimi et al. 2014]. In addition, solutions to potential resource saturation problems are being considered as future work activities, such as ALTO servers replying with round robin chosen versions

of the maps for equal cost paths to reduce the chances of multiple clients all choosing the same best destinations at once.

The intersection of ALTO with SDN-controlled domains are also an avenue of ongoing investigation as a means to facilitate inter-domain traffic engineering and SDN east/west interfaces. Implementing the remaining ALTO services (e.g., full Map-Service, EPS and ECS) is another open task, which adds to a number of performance optimization opportunities of our prototype, most of them related to Neo4j query tuning techniques (e.g., DB indexes, heap size, garbage collection and Linux fs configuration).

References

Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and Woundy, R. (2014). Application-Layer Traffic Optimization (ALTO) Protocol. RFC 7285.

Brito, S., Santos, M., Fontes, R., Lachos, D., and Rothenberg, C. (2015). Anatomia do Ecossistema de Pontos de Troca de Tráfego Publicos na Internet do Brasil. In XXXIII Simpósio Brasileiro de Redes de Computadores (SBRC). Vitória, ES, Brazil.

Choffnes, D. R. and Bustamante, F. E. (2008). Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 363–374. ACM.

Guanxiu, L., Suqi, Y., and Xinli, H. (2013). A Novel ALTO Scheme for BitTorrent-Like P2P File Sharing Systems. In *Proceedings of the 2013 Third International Conference on Intelligent System Design and Engineering Applications*.

Gurbani, V., Goergen, D., State, R., and Engel, T. (2014). Making historical connections: Building Application Layer Traffic Optimization (ALTO) network and cost maps from public broadband data. In *Network and Service Management (CNSM), 2014 10th International Conference on*.

Khan, A., Kwon, T., Kim, H.-c., and Choi, Y. (2013). AS-level Topology Collection Through Looking Glass Servers. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 235–242, New York, NY, USA. ACM.

Madhukar, A. and Williamson, C. (2006). A longitudinal study of p2p traffic classification. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 179–188.

Neo4j (2015). The Neo4j Manual v2.3.0-M03. http://neo4j.com/docs/milestone/.

Pinthong, N. and Lilakiatsakun, W. (2013). Performance of BitTorrent-like P2P file sharing systems inspired by ALTO. In *TENCON IEEE Region 10 Conference (31194)*.

Seedorf, J. and Burger, E. (2009). Application-Layer Traffic Optimization (ALTO) Problem Statement. RFC 5693.

Shibuya, M., Hei, Y., and Ogishi, T. (2011). ISP-friendly peer selection mechanism with ALTO-like server. In *Network Operations and Management Symposium (APNOMS)*.

Xie, H., Krishnamurthy, A., Silberschatz, A., and Yang, Y. R. (2007). P4p: explicit communications for cooperative control between p2p and network providers. *P4PWG Whitepaper*, pages 1–7.

Trilha Principal do SBRC 2016
Sessão Técnica 11
Distribuição de Vídeo

ESTRATÉGIA DE ADAPTAÇÃO DE FLUXO DE VÍDEO BASEADO EM FATORES DE QoE

Maiara de S. Coelho[1], **César A. V. Melo**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Caixa Postal 69077-000 – Manaus – AM – Brasil

{maiara,cavmelo}@icomp.ufam.edu.br

Abstract. *Due to increasing demand of HTTP Adaptive Streaming (HAS) in the last years, several studies have been performed to develop multimedia content distribution techniques to the Internet, aiming to improve the users Quality of Experience (QoE). Although there are a large number of studies in this area, many have not explored network resource sharing scenarios between multiple clients, making it necessary to study more about this complex scenario. This work presents a video stream adaptation strategy that takes into account factors related to QoE, as the playback interruption frequency. The goal is to minimize the amount of video playback interruption, thereby improving its continuity. The experiments are performed in extreme resource scarcity scenarios on a share across multiple clients. In the results, the used metrics are average video bit rate, interruptions, instability and fairness.*

Resumo. *A ampla adoção do Streaming Adaptativo sobre HTTP tem motivado estudos relacionados as técnicas de distribuição de conteúdo multimídia na Internet. Apesar da grande quantidade de estudos, poucos exploram cenários de compartilhamento da rede entre clientes, e o acoplamento entre a lógica de adaptação e o versionamento do conteúdo. Neste artigo apresenta-se uma estratégia de distribuição de conteúdo usando DASH que considera aspectos do versionamento na lógica de adaptação. Além disso, as decisões tomadas, para a adaptação do conteúdo as condições da rede, consideram fatores que afetam a Qualidade de Experiência do usuário. A estratégia proposta melhora a continuidade da sessão. Avaliou-se o desempenho da estratégia, considerando-se: número e duração das interrupções, taxa média de bits, instabilidade e injustiça, em diferentes cenários de rede, caracterizados pelo compartilhamento do canal de acesso por vários clientes.*

1. Introdução

Aplicações de *streaming* de vídeo têm recebido a atenção da comunidade que estuda o transporte e a entrega de conteúdo multimídia na Internet, face à crescente adesão as aplicações existentes, como YouTube e Netflix, e ao grande volume de tráfego gerado. Segundo perspectivas da Cisco [Cisco 2014], em 2019 o tráfego de vídeo responderá por 80% de todo o tráfego da Internet, equivalente a 89.319 PB/mês de um total de 111.592 PB/mês.

O *Streaming* Adaptativo sobre HTTP (do inglês: *HTTP Adaptive Streaming* - HAS) é a tecnologia de distribuição de *streaming* de vídeo que vem sendo crescentemente

empregada pelos distribuidores de conteúdo. O HAS consiste na disponibilização de cada vídeo em versões com diferentes níveis de qualidade, sendo que cada uma dessas versões é dividida em segmentos de igual duração. Isso permite que o cliente faça requisições individuais a cada segmento, alternando assim entre os níveis de qualidade a fim de adaptar o fluxo de dados às condições de rede e da aplicação. O *Streaming* Adaptativo Dinâmico sobre HTTP (do inglês: *Dynamic Adaptive Streaming over HTTP* - DASH) é uma das implementações do HAS, onde as versões são chamadas de representações. Além das representações, é disponibilizado um arquivo de Descrição da Mídia (do inglês: *Media Presentation Descriptor* - MPD) que descreve as informações sobre todas as versões de um determinado vídeo.

Definir as representações disponíveis no conjunto de versionamento de maneira otimizada é tão ou mais complexo quanto a adaptação da qualidade realizada pelo algoritmo adaptativo, pois é interessante que as representações estejam alinhadas ao perfil da rede de acesso do usuário. Um exemplo que ilustra essa complexidade são as possibilidades geradas pela combinação das diferentes plataformas de acesso e suas variações de configuração, incluindo resolução, largura de tela e tecnologias de acesso. Sendo assim, um mesmo versionamento pode não ser o mais adequado as redes cujas velocidades estão em perfis distintos, sobretudo, em cenários de compartilhamento entre clientes.

Neste artigo, propõem-se uma estratégia adaptativa para sistemas DASH, composta por um algoritmo de escalonamento e um versionamento particularizado. Existindo assim, um certo acoplamento entre o versionamento e o algoritmo. A estratégia, como um todo, objetiva a melhoria da continuidade de uma sessão de vídeo. Baseiando-se assim, em fatores importantes para o aumento da Qualidade de Experiência (do inglês: *Quality of Experience* - QoE) do usuário.

O restante do artigo está organizado da seguinte forma, os trabalhos relacionados são apresentados na Seção 2. Na Seção 3, descreve-se a arquitetura do sistema DASH utilizado e a estratégia proposta. Apresenta-se na Seção 4 a caracterização da coleção das métricas de avaliação, além dos resultados gerados nos experimentos conduzidos. Por fim, na Seção 5 apresentam-se as considerações finais.

2. Trabalhos Relacionados

O presente artigo está posicionado na área de transporte e entrega de conteúdo multimídia usando a tecnologia DASH. Assim, dois grupos de trabalhos foram considerados, as estratégias de adaptação e a preparação do conteúdo a ser distribuído.

As estratégias de adaptação podem ser agrupadas em três grandes classes: agressivas, conservadoras e moderadas. As **estratégias agressivas** utilizam todo o recurso estimado com base em medições instantâneas. As **estratégias conservadoras** usam os recursos estimados com uma margem de segurança, considerando que as estimativas são feitas com base em medições instantâneas. As **estratégias moderadas** representam uma abordagem intermediária a esses dois extremos, estimando os recursos disponíveis no sistema durante uma escala de tempo. A estratégia proposta neste artigo é moderada e por isso os trabalhos relacionados discutidos a seguir são aqueles identificados como tal.

As observações realizadas por uma estratégia moderada formam um histórico que é usado no processo de decisão realizado para definir a qualidade dos próximos segmentos

a serem acessados. Em [Cao et al. 2014], a escolha da qualidade do próximo segmento a ser acessado considera a estimativa da largura de banda, feita com a média aritmética da vazão dos últimos segmentos acessados em uma escala de tempo de 10 segundos. Além da estimativa da vazão, o crescimento inicial do *buffer* é usado para definir um estágio de execução rápida, permitindo que a taxa alvo seja alcançada rapidamente.

A estratégia apresentada em [Hesse 2013] utiliza uma média móvel como filtro de suavização para a estimativa de largura de banda. O estado do *buffer* também é considerado no cálculo do parâmetro de suavização utilizado pelo filtro. A estratégia proposta em [Jiang et al. 2014] calcula a média harmônica dos últimos 20 segmentos baixados, estimando assim a vazão da rede. Os autores ainda consideram o uso parcial do valor médio estimado para a tomada de decisões sobre a qualidade dos próximos segmentos baixados. Em [Juluri et al. 2015], uma estratégia similar é proposta com as seguintes variações: i) utiliza-se uma média harmônica e ii) a ponderação da média considera o tamanho dos segmentos em bytes.

Uma outra linha de trabalho propõem que a decisão do cliente seja complementada pelo operador da rede. A estratégia descrita em [Mok et al. 2012] controla a troca de qualidade a partir de um esquema de medição que considera a sondagem para a verificar a vazão com uma maior precisão. Em [El Essaili et al. 2015], evita-se a flutuação temporária da qualidade reescrevendo-se as requisições dos clientes a partir de um artefato instalado no provedor de acesso, que considera o conteúdo, no escalonamento da demanda, e os recursos disponíveis na rede.

O versionamento de conteúdo, embora seja parte importante das aplicações de vídeo adaptativo, tem poucos estudos sistemáticos registrados. Em [Toni et al. 2014], o versionamento de conteúdo foi estudado a partir da taxa de bits e da resolução demandada. Entretanto, não se considerou a continuidade e amplitudes das trocas de qualidade que o versionamento, feito de forma independente, pode impor as estratégias de adaptação.

Diferente dos trabalhos apresentados anteriormente, a estratégia proposta neste trabalho incorpora um nível mínimo de conhecimento do versionamento realizado. A estratégia usa esse conhecimento no seu processo de decisão, sem custos extras para a infraestratura de distribuição. Além disso, considera-se a frequência de interrupções, uma métrica de QoE apontada como chave [Seufert et al. 2015], para a tomada de decisão.

3. Estratégia de Adaptação Orientada a Métricas de QoE

Nesta seção, apresenta-se a estratégia de adaptação, chamada de QoE-Adapt, composta por algoritmo de adaptação e versionamento, que teve o seu espaço de projeto estabelecido pelas seguintes demandas:

- a redução das interrupções durante as sessões de reprodução, a partir do uso controlado de versões de baixa qualidade; e,
- o equilíbrio dos fatores de degradação da QoE com as demais métricas como taxa média de bits, instabilidade e injustiça.

Claramente, essas demandas estão em conflito mas quando controladas podem produzir sessões de vídeo de alta continuidade.

A Figura 1 mostra os elementos que compõem a estratégia proposta. Do lado do servidor DASH um conjunto de vídeos versionados, que possui, além de suas versões

padronizadas de acordo com o perfil da rede (detalhes na Seção 3.3), uma versão de qualidade inferior a mínima esperada para o perfil da rede. E, do lado do cliente DASH o mecanismo da estratégia de adaptação composta pelos seguintes elementos: i) o monitor de *buffer*, ii) o estimador de *buffer* e iii) o motor de adaptação.

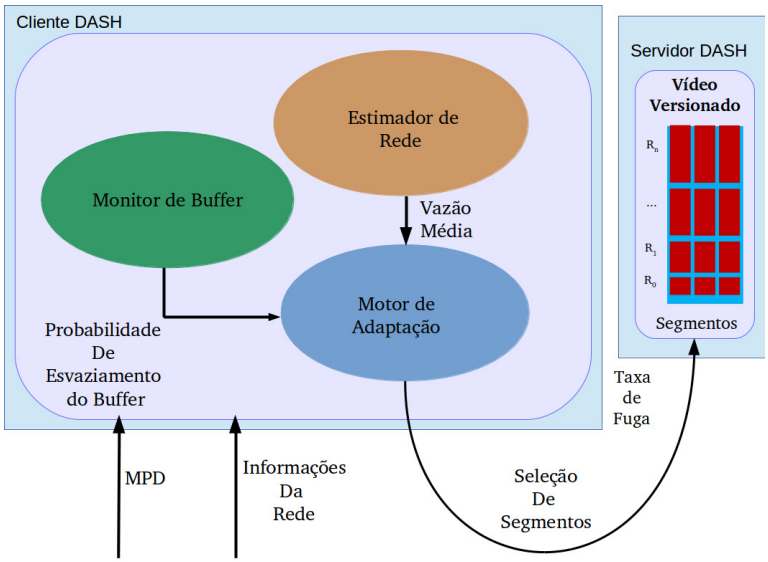


Figura 1. Módulos da Estratégia de Adaptação proposta

O monitor de *buffer* é responsável pelo cálculo da probabilidade de interrupção P da sessão pelo esvaziamento do *buffer*. O estimador de Rede calcula a vazão instantânea A e a vazão média $\hat{A}$, esta última considera os segmentos acessados em uma escala de tempo pré-estabelecida. Baseado nas informações dos dois primeiros módulos, o motor de adaptação decide qual a taxa de bits da representação R_{prox} do i-ésimo segmento que deve ser acessado. No caso da probabilidade P ter um valor maior que o limiar Θ, uma representação de qualidade inferior, chamada de taxa de fuga, será incluída no conjunto das representações acessíveis pela estratégia de adaptação.

O Algoritmo 1 detalha a lógica de adaptação. Como entrada, o algoritmo recebe o nível atual do *buffer* $B(t)$, a representação do último segmento baixado R_i, o limiar Θ da probabilidade de interrupção e o período de monitoramento do *buffer* Δ. O retorno é a representação do próximo segmento a ser baixado R_{prox}. A ocupação do *buffer* é caracterizada por três limiares: b_{min}, b_{low} e b_{max}. b_{min} é o limiar mínimo que sinaliza a iminência de uma interrupção, b_{low} é o limiar intermediário que indica a continuidade da reprodução nos próximos segundos, e b_{max} é o limiar máximo que indica a sobrecarga do *buffer*. No caso do b_{max} ser alcançado novas requisições estão suspensas até que o *buffer* alcançe nível de ocupação abaixo do seu limiar máximo. Nas linhas 1, 5 e 12, o algoritmo avalia o nível do *buffer*. A partir desse monitoramento, a largura de banda é calculada tendo-se como base os n segmentos baixados nos últimos Δ segundos (linha 2), ou ainda, no último segmento (linhas 6 e 9).

Quando o nível do *buffer* é menor ou igual ao limiar mínimo e a representação corrente é maior do que a representação da taxa de fuga (linha 12), avalia-se a frequência desse evento nos últimos Δ segundos (linha 13). Se essa frequência for superior ao limiar estabelecido (Θ) então está caracterizado um cenário de interrupção da reprodução, por

Algoritmo 1: Lógica de Adaptação - QoE-Adapt.

Entrada: $(B(t), R_i, \Theta, \Delta)$
Saída: (R_{prox})

```
1  se b_low < B(t) < b_max então
2      se Â(t − Δ) ∗ c > f(R_i) e R_i < R_max então
3          R_prox ← R_{i+1}
4  senão
5      se b_min < B(t) < b_low então
6          se A ∗ c < f(R_i) e R_i > R_min então
7              R_prox ← R_{i−1}
8          senão
9              se A ∗ c > f(R_i) e R_i < R_max então
10                 R_prox ← R_{i+1}
11     senão
12         se B(t) <= b_min e R_i > R_fuga então
13             se P(t − Δ) <= Θ então
14                 R_prox ← R_min
15             senão
16                 R_prox ← R_fuga
```

isso o próximo segmento a ser acessado no servidor será de uma representação com taxa de bits igual a taxa de fuga. Caso contrário, a taxa mínima desejada é usada como a taxa do próximo segmento a ser baixado.

3.1. Estimativa de Largura de Banda Disponível

O módulo estimador de rede determina a vazão instantânea A e a vazão média $\hat{A}$. A é dada pela razão entre o tamanho do último segmento e o tempo de *download* do segmento, $\hat{A}$ é calculada utilizando uma média móvel exponencial da vazão A dos segmentos acessado nos últimos Δ segundos. A Equação 1 mostra o cálculo da vazão média:

$$\hat{A} = \begin{cases} \delta\hat{A}(i-1) + (1-\delta)A(i), & \text{se} \quad i > 1 \\ A(1), & \text{se} \quad i = 1 \end{cases} \tag{1}$$

onde, δ é o peso utilizado para ponderar os valores de vazão.

3.2. Frequência relativa de Interrupções de Reprodução

A interrupção da reprodução é o fator mais impactante na QoE do usuário, pois reflete diretamente na continuidade da sessão. Neste trabalho, caracterizou-se a iminência de uma interrupção a partir da ocupação do *buffer*. Quando essa ocupação atinge valores menores que o limiar mínimo b_{min}, um certo número de vezes, dentro de uma escala de tempo de Δ segundos, fica caracterizada a iminência da interrupção.

A estratégia proposta além de apresentar características de uso moderado dos recursos, antecipa a tomada de decisão, frente a iminência do evento de interrupção, caracterizando-se também como proativa. Na Equação 2 apresenta-se o cálculo da probabilidade de interrupções baseado nos níveis de *buffer* registrados nos últimos Δ segundos:

$$P(E) = \frac{N(E)}{N(S)} \tag{2}$$

onde, $N(E)$ é o número de vezes que o nível do *buffer* esteve abaixo do seu limiar mínimo e $N(S)$ é o número total de observações feitas ao *buffer*.

A ideia central das dinâmicas de avaliação da largura de banda e os níveis de *buffer* foram originalmente descritos em [Akhshabi et al. 2012], as quais foram implementadas e adaptadas no *player* [DASHIF 2015] utilizado neste trabalho. Essa lógica de adaptação é considerada como a base para os estudos comparativos realizados neste artigo e seu desempenho é identificado sob o nome de Adaptech.

3.3. Versionamento

O vídeo utilizado no estudo está disponível em [Sintel 2013]. O conjunto de representações originado nesse processo limita a ação do algoritmo adaptativo, pois é dentro do espaço de possibilidades oferecidas por esse conjunto que evoluirá a lógica da estratégia de adaptação. O cenário torna-se mais desafiador quando o canal de acesso é compartilhado por diversos usuários, dado que nesses casos é provável que hajam maiores variações na capacidade de transferência do canal, gerando descompasso entre a taxa de transferência e as taxas disponíveis no conjunto de representações originalmente planejado.

Em [Coelho 2015], estudo que gerou o presente artigo, estão descritas detalhadamente todas as decisões de projeto e estudos comparativos que influenciaram nestas decisões. Algumas das decisões mais importantes foram quanto aos valores das taxas e a granularidade (espaçamento) entre essas taxas nos conjuntos de versionamento. Quanto as taxas, foram padronizadas dentro de uma faixa de valores correspondentes a adesão dos domicílios brasileiros as redes de acesso, registrada pelo relatório do Cetic, período de 2011 a 2015 [Cetic.Br 2015], com exceção da taxa de fuga. Quanto a granularidade, o estudo apontou que alta granularidade gera um menor número de representações e isso reflete diretamente na velocidade da convergência para a taxa alvo.

Desta forma, foram utilizados três tipos de versionamento: o primeiro seguiu a recomendação da Netflix que demanda uma taxa mínima de 1000 kbps e uma taxa máxima de 3600 kbps; o segundo versionamento seguiu as recomendações geradas pelo próprio estudo comparativo e possui a taxa mínima de 1600 a taxa máxima de 3660 kbps, com granularidade de taxa de bits igual a 400 kbps. O terceiro versionamento, segue o estabelecido no segundo, com a exceção da inclusão de uma taxa de menor qualidade, chamada taxa de fuga, igual a 460 kbps que é usada em conjunto com o algoritmo adaptativo e caracteriza a estratégia proposta neste artigo.

Utilizou-se o FFmpeg para o versionamento dos vídeos e o MP4Box para a segmentação e a geração do MPD. A duração dos segmentos é de 5 segundos.

4. Experimentos

Nesta seção apresentam-se aspectos considerados na coleta dos dados, as métricas de avaliação, a configuração do ambiente de experimentação e, por fim, os resultados dos experimentos realizados para avaliar a estratégia de adaptação proposta. Os resultados incluem estudos comparativos de desempenho da estratégia proposta com uma estratégia moderada, chamada Adaptech, sob diferentes circunstâncias de versionamento e disponibilidade de recursos na rede.

4.1. Coleta dos Dados

O cenário criado para a experimentação das diferentes estratégias e condições de rede demandou a implementação de um serviço que permite o armazenamento dos dados em uma base de dados. Primeiramente os dados são armazenados em uma estrutura linear no cliente DASH, durante a sessão de vídeo, e ao fim da sessão o cliente envia uma mensagem do tipo POST, para o servidor, contendo em seu corpo um arquivo JSON com todas as informações de largura de banda, do *buffer* e dos segmentos. Tais dados são posteriormente utilizados nos cálculos das métricas de desempenho das estratégias.

As informações referentes aos segmentos e a vazão são: data/hora, data/hora da requisição, data/hora do início do *download* e data/hora de finalização do *download*, tamanho do segmento em bits, duração do segmento em segundos, número da representação, taxa de bits em kbps, vazão em kbps. Todas as data e horas são sincronizadas ao relógio do servidor, indicando o tempo corrente que será confrontado com o tempo de inicio da sessão de acesso. As informações referentes ao *buffer* são: data/hora e nível de ocupação do *buffer* em segundos. Novamente, a informação temporal considera o tempo registrado no servidor e o tempo de inicio da sessão de acesso. As informações referentes ao experimento realizado são estas: identificação do teste, data/hora inicial da sessão de vídeo, data/hora final da sessão de vídeo, tipo de fluxo - áudio ou vídeo, algoritmo de adaptação de taxa que está sendo testado, endereço do MPD e número do cenário utilizado.

4.2. Métricas de Avaliação

As métricas utilizadas para avaliar a estratégia de distribuição são: o número de interrupções, a duração das interrupções, a taxa de bits média, a instabilidade e a injustiça.

O Número de Interrupções indica o número de vezes em que o *buffer* esvaziou-se e o vídeo parou sua reprodução, provocando o congelamento da imagem do vídeo. A **Taxa de Bits Média** μ, definida em [Mueller et al. 2012], é dada pela seguinte Equação:

$$\mu = \frac{\sum_{i=0}^{n} (f(i) * D_i)}{T_s} \tag{3}$$

onde, i representa o i-ésimo segmento, $f(i)$ é a função que retorna a taxa de bits do segmento i, D_i representa a duração do segmento i e Ts representa a duração da sessão.

A **Instabilidade** apresentada em, [Jiang et al. 2014], expressa a soma dos pesos da amplitude das trocas de qualidade ao longo da reprodução do vídeo. E de todas as trocas dos últimos k segmentos observados, dividido pela soma de todas as taxas de bits acessadas com seus pesos. A Equação 4 apresenta o cálculo da instabilidade:

$$S = \frac{\sum_{i=0}^{k-1} \left(|b_{x,t-i} - b_{x,t-i-1}| * \omega(i) \right)}{\sum_{i=1}^{k} \left(b_{x,t-i} * \omega(i) \right)} \tag{4}$$

onde, $\omega(i) = k - i$ é uma função de penalidade linear que aloca maior peso para trocas mais recentes. Esse peso é agregado também as taxas de bits acessadas, k representa a quantidade de segmentos passados que devem ser avaliados.

O **Índice de Justiça** [Jiang et al. 2014], expressa o quão igualitário é um algoritmo dentro de um cenário de compartilhamento. O índice de injustiça é definido pela seguinte equação:

$$I = \sqrt{\frac{[\sum_{i=1}^{n} (x_i)]^2}{\sum_{i=1}^{n} (x_i)^2}} \tag{5}$$

onde, x_i representa a razão do recurso esperado pelo recurso obtido, por recurso esperado entende-se a média de taxa de bits esperada para cada sessão do compartilhamento e recurso obtido a taxa média de bits de fato acessada em cada cliente.

4.3. Configuração do Ambiente de Experimentação

A Figura 2 apresenta o ambiente de experimentação. Onde a vazão da rede foi controlada pelo DummyNet, conforme o cenário de rede projetado em cada experimento. Todo o tráfego que chega e sai do servidor foi capturado pelo Wireshark, e posteriormente avaliado.

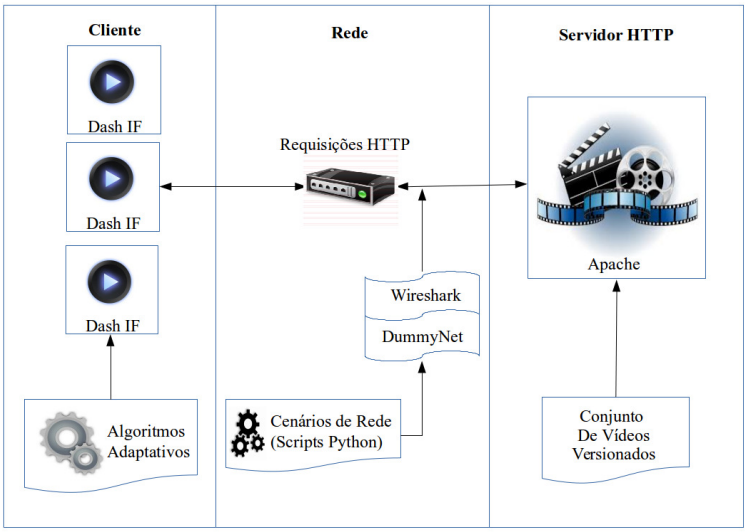


Figura 2. Ambiente de experimentação com compartilhamento de recursos.

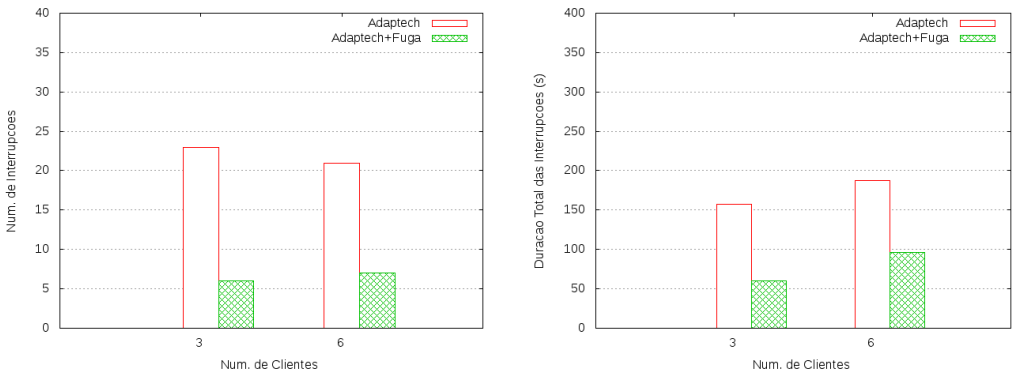
Considerou-se cenários de rede com três e seis clientes compartilhando um canal com taxa de 3Mbps e 5Mbps, respectivamente. Essas valores foram definidos com base na adesão as redes acesso fixa, apresentada em [Cetic.Br 2015], onde mostra uma crescente adesão as redes de mais alta taxa, que operam com taxas de até 8 Mbps e acima de 8 Mbps. Nestes grupos de mais alta taxa, aplicou-se a regulamentação da Anatel que estabelece que os provedores devem garantir o mínimo de 40% da taxa contratada, tendo-se considerado

portanto larguras de banda máximas de 7,5 Mbps e 12,5 Mbps. É importante observar que a variação da largura de banda se dá pelo compartilhamento do canal.

4.4. Resultados Numéricos

Os resultados apresentados a seguir foram obtidos durante a execução de 182 sessões de transmissão de um vídeo, que tem duração de 14 minutos. Nesse cenário as métricas foram calculadas somente no intervalo de tempo em que os clientes acessam simultaneamente o conteúdo, sendo apresentado aqui o valor médio obtido das sessões.

O primeiro estudo apresentado considera o impacto da taxa de fuga na continuidade da sessão de vídeo, quando a estratégia base Adaptech gerência a sessão, Figura 3. Foram medidos o número de interrupções, Figura 3(a), e a duração total das interrupções, Figura 3(b). Independente do número de clientes compartilhando o canal, verifica-se uma redução significativa na métrica número de interrupções. No cenários de compartilhamento de três e seis clientes, a redução no número de interrupções foi, respectivamente, de 73,91% e 51,85%.



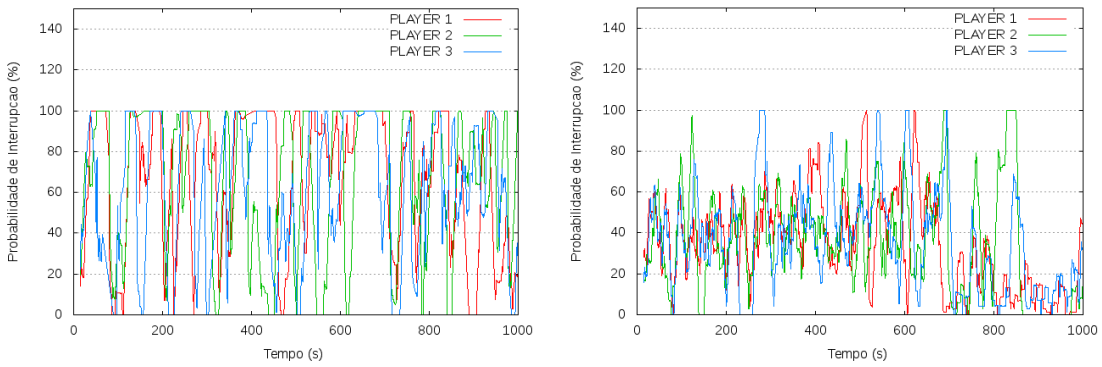
(a) Número de Interrupções (b) Duração Total das Interrupções

Figura 3. A Estratégia Adaptech com Versionamento Sem Taxa de Fuga e Com Taxa de Fuga, Número e Duração Total das Interrupções por Número de Clientes.

Outro fator importante para a avaliação da QoE é a duração das interrupções. As interrupções prolongam a sessão de acesso, impactando de forma negativa na retenção da audiência. A Figura 3(b) apresenta o aumento médio das sessões provocado pelas interrupções quando um canal é compartilhado por três e seis clientes. A presença da taxa de fuga permitiu a estratégia Adaptech reduções significativas na duração da sessão. Quando apenas três clientes compartilham o canal essa redução foi de 61,78% e, quando seis clientes compartilham o canal tal redução atingiu o valor de 48,71%.

Avaliou-se também de que forma a presença de uma taxa de fuga impacta na frequência com que o *buffer* atinge uma ocupação inferior a mínima esperada. Na Figura 4 apresenta-se a evolução da frequência relativa das interrupções ao longo do tempo. A iminência de interrupções é muito mais frequente quando a taxa de fuga não está presente no conjunto de representações disponíveis, Figura 4(a). Quando a Taxa de fuga é utilizada (Figura 4(b)) verifica-se uma mudança na concentração dos eventos de esvaziamento, que ficam em torno de 40% a 60%, havendo assim, uma nítida suavização das vezes em que a probabilidade alcança seu valor máximo. O intervalo [40%, 60%] constitui o espaço de

otimização do limiar que caracteriza a iminência das interrupções. É importante observar que a quantidade de interrupções (Figura 3) para o versionamento sem taxa de fuga, está refletida em um número de vezes maior com que a frequência relativa das Interrupções é igual a 1.0 (Figura 4).



(a) Adaptech (b) Adaptech+Fuga

Figura 4. Iminência de interrupções ao longo do Tempo.

A utilização da taxa de fuga permitiu uma maior continuidade de reprodução, entretanto o acesso por períodos longos a representação com baixa qualidade afeta outras métricas, i.e. taxa média de bits. O acesso a representações de menor qualidade é resultado em parte das condições extremas a qual a rede foi submetida, mas também do comportamento moderado do algoritmo base, que demanda condições certamente favoráveis da rede para realizar acesso a representações com melhor qualidade. Esse cenário demandou as melhorias implementadas no algoritmo base, que consistiu na caracterização do cenário de rede em que o uso da taxa de fuga é mandatório. Os resultados dessa caracterização e seu uso são apresentados nas avaliações que seguem.

Na Figura 5 apresenta-se a taxa média de bits registrada quando três e seis clientes, estão acessando o mesmo vídeo. A conclusão geral é que independente do versionamento empregado, sob condições extremas de uso, a taxa média de bits será menor que a mínima esperada, que é de 1000 kbps para o versionamento sugerido pelo NetFlix e 1600 kbps sugerido por este estudo. O aumento da duração das sessões e impossibilidade de acessar taxas mais altas, devido a escassez de recursos, são o motivo para esse resultado.

Um ponto a ser destacado é que a estratégia QoE-Adapt equipara-se ao melhor resultado da estratégia Adaptech, apresentando, como será mostrado a seguir, um número bem menor de interrupções. Em relação a estratégia Adaptech com versionamento sugerido pelo Netflix, Adaptech-Netflix, a estratégia QoE-Adapt tem desempenho 35,8% e 79,4% melhor, para cenários com três e seis clientes, respectivamente.

Na Figura 6 apresentam-se as medições realizadas para o número de interrupções e para a duração das interrupções com as três estratégias de acesso ao conteúdo e cenários de rede com três e seis clientes. Os resultados mostram que o desempenho da estratégia QoE-Adapt é melhor que aquele verificado pela estratégia Adaptech e Adaptech+Netflix. Na comparação entre as estratégias QoE-Adapt e Adaptech a redução nas interrupções foi de 43,47% em ambos os cenários de rede. Ao comparar as estratégias QoE-Adapt e Adaptech-Netflix verifica-se redução a favor da estratégia QoE-Adapt de 59,25% e

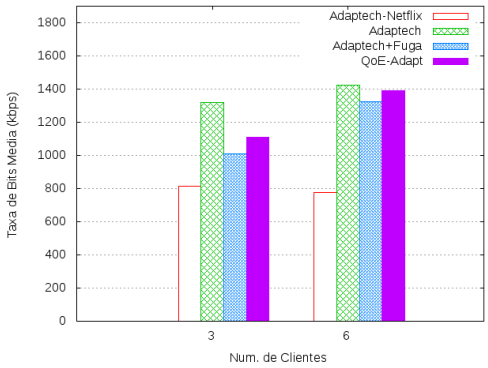
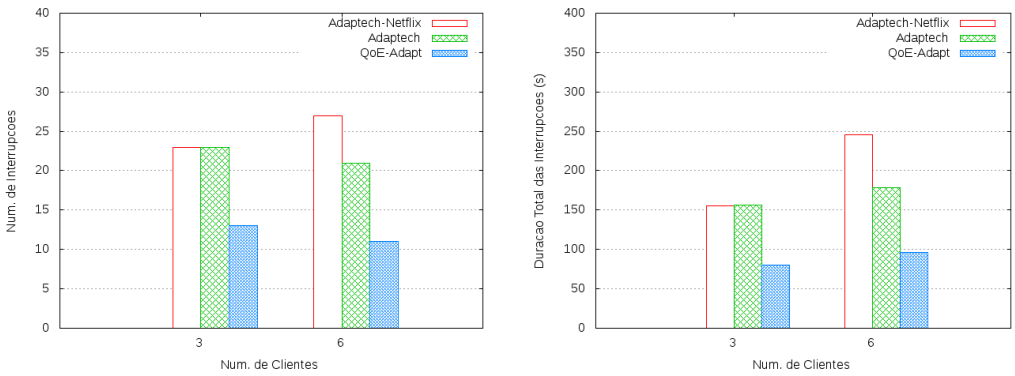


Figura 5. Taxa de Bits Média por Número de Clientes.

47,61% no mesmo cenário de rede.

Com relação a duração das interrupções, verificou-se que a estratégia QoE-Adapt apresenta o melhor desempenho entre as três avaliadas. Esse desempenho foi quantificado por uma redução na duração das sessões em 48,33% e 48,72%, em relação a estratégia Adaptech-Netflix e de 60,63% e 45,99%, em relação a estratégia Adaptech, quando três e seis clientes compartilhavam o canal.

O acesso controlado a taxa de fuga reduz a influência dos aspectos mais conservadores que conduzem a um número menor de acesso as taxas de bits mais altas, que ainda está presente na estratégia QoE-Adapt. Por outro lado, identifica-se que essa mesma ação gera um equilíbrio maior entre as outras métricas e a taxa de bits média, que, conforme discutido anteriormente, apresenta valores melhores que o verificado quando não há restrição de acesso a taxa de fuga.



(a) Número de Interrupções (b) Duração Total das Interrupções

Figura 6. Número e Duração Total das Interrupções por Número de Clientes.

A instabilidade de um algoritmo é afetada pelas características da estratégia. Uma abordagem conservadora busca representações de melhor qualidade somente quando as condições da rede são claramente favoráveis, enquanto que em abordagens mais agressivas condições mínimas de rede já são suficientes para se buscar representações de melhor qualidade. Esses cenários são potencializados por possível descompasso entre o versionamento e a capacidade do canal de acesso. A Figura 7 apresenta a instabilidade

registrada para as estratégias Adaptech-Netflix, Adaptech e QoE-Adapt. A estratégia QoE-Adapt apresenta valores altos para a instabilidade (0,353) quando comparado com as estratégias Adaptech-Netflix (0,074) e Adaptech (0,038). Esse comportamento deve-se ao acesso controlado a taxa de fuga presente na estratégia QoE-Adapt. Por se tratar de uma representação com taxa de bits bem menor que a mínima desejada, a amplitude de uma troca envolvendo a taxa de fuga, contribui de forma significativa para o aumento da instabilidade, ver Equação 4.

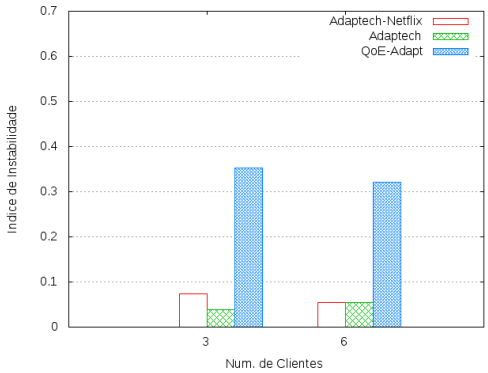


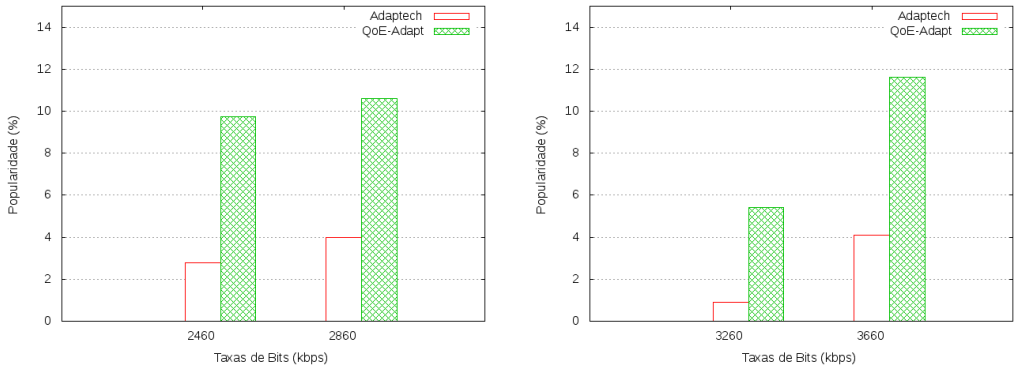
Figura 7. Índice de Instabilidade por Número de Clientes.

Além disso, ao dispor de uma representação com taxa de bits menor que a vazão da rede, a estratégia QoE-Adapt abre a possibilidade de alívio das condições de congestionamento permitindo que em momentos oportunos haja acesso com maior frequência as representações com taxas de bits mais altas. Essa dinâmica gera ainda um efeito apreciado pelos usuários que é a redução de ocorrência de instabilidade de amplitude negativa, que se dá quando a representação escolhida tem taxa de bits menor que a última representação acessada, e aumenta a ocorrência de instabilidade por amplitude positiva, que se traduz em melhoria da qualidade do vídeo.

O aumento desse tipo de ocorrência permite que as representações com taxas de bits maiores sejam acessadas por mais tempo, considerando o tempo total da sessão. Na Figura 8, apresenta-se a popularidade das taxas mais altas acessadas pelos algoritmos, em cenários de rede com três e seis clientes. A conclusão geral é que a estratégia QoE-Adapt promove uma quantidade maior de acesso as taxas de bits mais altas. No cenário de rede com três clientes, Figura 8(a), o uso da estratégia QoE-Adapt aumentou o acesso as taxas mais altas de 5% à 17%, e no cenário com seis clientes, Figura 8(b), aumentou de 6,8% para 20,35%.

A estratégia Adaptech apresenta baixa instabilidade devido a inexistência de condições favoráveis para acesso as representações com taxas de bits mais alta, e, por outro lado, o descompasso entre a vazão da rede e a taxa de bits mínima desejável, obriga a estratégia a permanecer acessando essa taxa impedindo que se registre uma maior ocorrência de instabilidade de amplitude negativa.

O índice de injustiça, conforme discutido na Seção 3.3, mede quão igualitário é um algoritmo no compartilhamento de recursos. A Figura 9 mostra os resultados coletados para essa métrica. Nos cenários de menor escassez de recursos as estratégia se equiparam, entretanto quando a escassez de recursos aumenta o índice de injustiça da estratégia QoE-



(a) Utilizando 3 Clientes (b) Utilizando 6 Clientes

Figura 8. Popularidade das Taxas de Bits mais Altas.

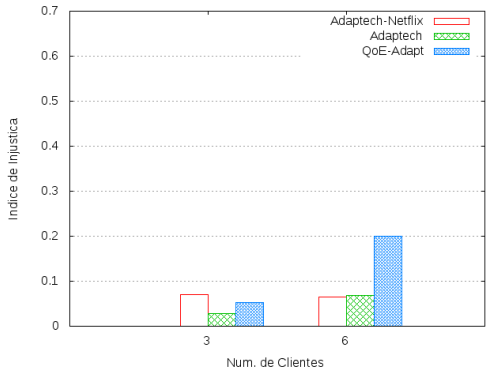


Figura 9. Índice de Injustiça por Número de Clientes.

Adapt se destaca. Nesse cenário, ambas estratégias Adaptech e Adaptech-Netflix foram forçadas a permanecer na mesma taxa em função da escassez de recursos. Por outro lado, a QoE-Adapt ao fazer uso da taxa de fuga, permite que a rede tenha uma redução de demanda, impulsionando assim o acesso a taxas de bits mais altas por outros clientes. Em outras palavras, uma injustiça alta em cenários extremos de compartilhamento não deve ser traduzido diretamente como característica indesejável da estratégia.

5. Conclusão

Neste artigo apresentou-se uma estratégia capaz de adaptar as taxas de bits de acordo com as condições da rede, levando em consideração a probabilidade de interrupções na reprodução do vídeo.

Os estudos numéricos conduzidos consideraram diferentes métricas quantitativas, i.e. número e duração das interrupções, taxa de bits média, e métricas qualitativas, i.e. instabilidade e justiça.

A estratégia proposta alcançou o objetivo de diminuição das interrupções. Uma investigação maior se faz necessária no sentido incluir outros fatores que impactam a QoE, e.g. número de trocas de positivas, no conjunto de possíveis variáveis a serem consideradas no processo de decisão.

Referências

Akhshabi, S., Narayanaswamy, S., Begen, A. C., and Dovrolis, C. (2012). An experimental evaluation of rate-adaptive video players over http. *Signal Processing: Image Communication*, 27(4):271–287.

Cao, Y., You, X., Wang, J., and Song, L. (2014). A qoe friendly rate adaptation method for dash. *Broadband Multimedia Systems and Broadcasting (BMSB), 2014 IEEE International Symposium on*, pages 1–6.

Cetic.Br (2015). Centro de estudos sobre as tecnologias da informação e da comunicação. http://cetic.br/pesquisa/domicilios/indicadores.

Cisco (2014). Cisco visual networking index: Forecast and methodology, 2014-2019.

Coelho, M. (2015). Estratégia de adaptação de fluxo de vídeo baseada em fatores de qoe. Master's thesis, Universidade Federal do Amazonas, UFAM.

DASHIF (2015). Reference client. http://dashif.org/reference/players/javascript/index.html.

El Essaili, A., Schroeder, D., Steinbach, E., Staehle, D., and Shehada, M. (2015). Qoe-based traffic and resource management for adaptive http video delivery in lte. *Circuits and Systems for Video Technology, IEEE Transactions on*, 25(6):988–1001.

Hesse, S. (2013). Design of scheduling and rate-adaptation algorithms for adaptive http streaming. *SPIE Optical Engineering+ Applications*, pages 88560M–88560M.

Jiang, J., Sekar, V., and Zhang, H. (2014). Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *Networking, IEEE/ACM Transactions on*, 22(1):326–340.

Juluri, P., Tamarapalli, V., and Medhi, D. (2015). Sara: Segment aware rate adaptation algorithm for dynamic adaptive streaming over http. *IEEE ICC Workshop on QoE-Management for Duture Internet Applications and Services - QoE-FI*.

Mok, R. K. P., Luo, X., Chan, E. W. W., and Chang, R. K. C. (2012). Qdash: A qoe-aware dash system. *Proceedings of the 3rd Multimedia Systems Conference*, pages 11–22.

Mueller, C., Lederer, S., and Timmerer, C. (2012). An evaluation of dynamic adaptive streaming over http in vehicular environments. *Proceedings of the 4th Workshop on Mobile Video*, pages 37–42.

Seufert, M., Egger, S., Slanina, M., Zinner, T., Hobfeld, T., and Tran-Gia, P. (2015). A survey on quality of experience of http adaptive streaming. *Communications Surveys Tutorials*, (1):469–492.

Sintel (2013). Durian open movie project. http://durian.blender.org/.

Toni, L., Aparicio-Pardo, R., Simon, G., Blanc, A., and Frossard, P. (2014). Optimal set of video representations in adaptive streaming. *Proceedings of the 5th ACM Multimedia Systems Conference*, pages 271–282.

A Rede tem a Resposta: Um Modelo Escalável para Predição Integrada de Qualidade de Vídeo e QoE em Redes IP

Roberto Irajá Tavares da Costa Filho[1], **William Lautenschläger**[1], **Hugo Schroter Lazzari**[1], **Valter Roesler**[1], **Luciano Paschoal Gaspary**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{roberto.costa,wrlautenschlager,hslazzari,roesler,paschoal}@inf.ufrgs.br

Abstract. *Due to the fact that video streaming is the current "killer"application and to competiveness, telecommunication service providers need to be able to answer a fundamental question: to which extent is the available network infrastructure able to successfully provide users with a satisfactory experience when running video streaming applications? Answering this question is far from trivial, because existing techniques are neither scalable nor accurate enough. To address this issue, we propose a model to predict video streaming quality based on the observation of performance indicators of the underlying IP network. To accomplish this objective, the proposed model — created using LTE networks as case study — leverages low network consumption active measurements and machine learning techniques. Obtained results show that the proposed solution produces accurate estimates (average error of less than 10%) while keeps intrusiveness around twenty times lower than traditional techniques.*

Resumo. *Pelo fato de streaming de vídeo representar aplicação "matadora"e por competitividade, provedores de serviços de telecomunicações precisam ser capazes de responder uma pergunta fundamental: em que medida a infraestrutura de rede disponível tem dado conta de oferecer aos usuários uma percepção de qualidade satisfatória quando empregam essas aplicações? Obter resposta a essa pergunta está longe de ser trivial, pois as técnicas existentes ou não são escaláveis ou são pouco acuradas. Para suprir essa lacuna, propõe-se um modelo de predição de qualidade de streaming de vídeo com base na observação de desempenho da rede IP subjacente. Para tal, o modelo proposto — criado usando redes LTE como estudo de caso — lança mão de técnica de medições ativas de baixo consumo de banda e de aprendizagem de máquina. Os resultados alcançados evidenciam que a solução proposta produz estimativas acuradas (erro médio abaixo de 10%) e apresenta intrusividade cerca de vinte vezes menor que técnicas tradicionais.*

1. Introdução

A taxa de penetração da banda larga móvel estimada pelo ITU para 2014 foi de 83,7% em países desenvolvidos, enquanto que os acessos de banda larga fixa, no mesmo período, chegaram a 27,5% da população [ITU 2015]. Em paralelo com o aumento do número de acessos, a proliferação das aplicações de vídeo aumenta a complexidade da tarefa de gerenciar a qualidade dos serviços providos, pois a aplicação de vídeo, quando comparada a outras aplicações, exige uma grande quantidade de recursos da rede. Estima-se que até 2016 o tráfego de vídeo na Internet represente cerca de 86% do volume total cursado na rede [Katsarakis et al. 2014].

Nesse cenário, tanto a comunidade científica quanto a indústria concordam que maximizar a qualidade de experiência do usuário (QoE), quando do uso de aplicações de *streaming* de vídeo, representa um desafio de pesquisa relevante [Katsarakis et al. 2014] [Quadros et al. 2015]. Um aspecto chave nessa direção consiste em aferir, de forma sistemática, a qualidade dos serviços de vídeo providos. O ambiente de operadora exige uma solução de baixa intrusividade, que permita medições em larga escala, e que mensure a qualidade do serviço de vídeo entregue ao usuário com um grau aceitável de acurácia. Essa tarefa torna-se mais complexa ao se considerar redes celulares, foco deste artigo, em que medições com maior intrusividade têm potencial para interferir diretamente na qualidade dos serviços providos.

Um caminho possível para resolver o problema foi estudado por Msakni e Youssef [Msakni and Youssef 2013]. Eles realizaram uma análise sobre técnicas propostas recentemente para predição de QoE que dispensam a necessidade de transferir vídeos para conduzir medições. O estudo concluiu que, até o momento, não existe uma técnica confiável para realizar tal estimativa. Em essência, a não linearidade da opinião humana acaba por comprometer a acurácia do uso de indicadores de camada de rede como preditores *diretos* de QoE, pois um mesmo comportamento de rede poderia ser mapeado para notas distintas de MOS (*Mean Opinion Score*). Um outro caminho possível consiste em avaliar a qualidade de vídeos de forma objetiva junto ao usuário. No entanto, as técnicas que permitem tal medição implicam na necessidade de transferir e analisar arquivos reais de vídeo, incorrendo em acréscimo substancial de tráfego na rede.

Neste artigo propõe-se um modelo para predição de indicadores de desempenho para o serviço de *streaming* de vídeo por meio da observação de indicadores de desempenho da rede IP subjacente. O modelo proposto explora a técnica de árvores de decisão para estabelecer e quantificar a relação entre indicadores de QoS[1] e indicadores objetivos que infiram a qualidade da reprodução de vídeos no contexto do usuário final, doravante denominados AppQoS[2]. Adicionalmente, por meio de um processamento de AppQoS, o modelo proposto permite inferir a qualidade da experiência do usuário. Nesta primeira iteração para resolver o referido problema, focou-se — como estudo de caso — em redes LTE. Resultados obtidos a partir de avaliação em um ambiente controlado permitem observar que o modelo proposto é adequado para viabilizar predição em larga escala da qualidade dos serviços de vídeo entregue aos usuários, com baixa intrusividade.

O restante do artigo está organizado da seguinte forma: a seção 2 aborda os trabalhos relacionados. A seção 3 apresenta o modelo de predição proposto. Na seção 4 aborda-se a configuração do ambiente experimental e os aspectos de construção do modelo. Na seção 5 é relatada a avaliação de desempenho, bem como a aplicação do modelo, enquanto a seção 6 conclui o artigo e apresenta perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

Mecanismos clássicos como MOS, PEVQ (*Perceptual Evaluation of Video Quality*) e PSNR (*Peak Signal-to-Noise Ratio*) são reconhecidamente não escaláveis quando utilizados para determinar — de forma sistemática — se uma infraestrutura de rede está (ou não) prestando o suporte necessário a aplicações de vídeo. Tal se deve por requererem entrevistas com usuários e/ou demandarem transferências de vídeos. Na tentativa de contornar tais limitações, mais recentemente foram conduzidos trabalhos de pesquisa com o objetivo de permitir inferência de qualidade, mas calcados em procedimentos de aquisição de

[1]O termo indicadores de QoS é empregado para referenciar indicadores de desempenho de redes IP.

[2]O termo AppQoS é empregado por representar a última barreira objetiva possível de ser medida no contexto do usuário final.

dados que não tornem o seu uso proibitivo para ambientes de larga escala. Nesse contexto, dois grupos de trabalhos relacionados — doravante denominados grupo A e grupo B — se destacam. De forma geral, trabalhos que se encaixam no grupo A caracterizam-se por, partindo de indicadores de QoS, estimarem diretamente QoE. Por não levarem em conta indicadores de desempenho objetivos da aplicação, não permitem capturar fidedignamente se a rede está oferecendo o suporte adequado à aplicação e acabam sujeitos a estimativas influenciadas pela não linearidade da opinião humana. Por sua vez, trabalhos pertencentes ao grupo B preocupam-se em estabelecer relação entre indicadores de camada de aplicação e QoE, mas sem vinculá-los com indicadores de QoS. Tal não ajuda o provedor de serviço a entender a influência da rede nos indicadores de QoE.

Como um exemplo representativo de trabalho de pesquisa do grupo A, De Pessemier *et al.* [De Pessemier et al. 2013] avaliaram a influência da variação de parâmetros de QoS da rede na qualidade de experiência de usuários para aplicações de vídeo que utilizam DASH (*Dynamic Adaptive Streaming over HTTP*). Os autores realizaram experimentos em que parâmetros como atraso e vazão da rede foram degradados de forma controlada. Ao mesmo tempo, mediram por meio de MOS a satisfação dos usuários quanto à qualidade da reprodução. Embora considere e manipule parâmetros de QoS, o trabalho está focado no entendimento da tolerância do usuário em relação à qualidade de vídeos, ou seja, QoE, e não na caracterização da influência de parâmetros de QoS na qualidade objetiva da reprodução de vídeos.

Ainda como trabalho do grupo A, Hsu e Lo [Hsu and Lo 2014] propuseram um modelo de mapeamento entre QoS e QoE para o contexto de uma rede *multicast* em nuvem. Nesse caso, os autores realizaram uma série de experimentos para estabelecer a relação entre os parâmetros de QoS configurados na rede *multicast* e o seu respectivo impacto na nota MOS atribuída pelos usuários ao assistirem vídeos. Embora resolva o problema de escalabilidade via uso de um preditor, a proposta converge em um indicador impreciso para provedores de serviço, pois uma mesma condição de rede, em razão da análise subjetiva, pode ser mapeada em diferentes notas de MOS.

O grupo B reúne pesquisas orientadas ao estudo dos fenômenos que o uso do protocolo HTTP introduz na reprodução de vídeos, bem como o seu reflexo na experiência do usuário. Para mensurar os principais fenômenos foram estabelecidos indicadores, dos quais pode-se destacar: atraso de inicialização na reprodução de um vídeo, número e duração de interrupções de reprodução causadas por esvaziamento de *buffer* e somatório de tempos de parada [Balachandran et al. 2012].

Na mesma direção, o trabalho desenvolvido por Balachandran *et al.* [Balachandran et al. 2013] introduz uma técnica para predição de QoE por meio da análise de indicadores de desempenho da camada de aplicação. Para tanto, os autores consideraram uma base de indicadores de camada de aplicação já existente sobre o comportamento do *player* de usuários reais. Para obtenção desses indicadores, esse tipo de técnica necessita que o usuário tenha um *plugin* instalado no seu dispositivo de acesso e que os vídeos sejam efetivamente reproduzidos. Embora os indicadores produzidos sejam valiosos para que o provedor possa estimar QoE, essa informação é incompleta na medida em que não permite ao provedor compreender a influência do desempenho de sua rede na qualidade de experiência do usuário.

Nesse cenário, o presente trabalho avança no estado da arte na medida em que, diferente dos trabalhos dos grupos A e B, propõe o uso de indicadores de QoS para estimar indicadores intermediários entre QoS e QoE, os quais rotulou-se AppQoS. Além disso, de posse de indicadores de AppQoS é possível inferir a qualidade da experiência do usuário

de forma indireta. Dessa forma, o indicador de QoE permite destacar de forma preliminar os pontos na rede que estão sofrendo degradação de qualidade percebida, enquanto que os indicadores de AppQoS fornecem insumos que permitem ao provedor compreender as causas da queda de qualidade, bem como fornecem subsídios para avaliar a efetividade de eventuais intervenções na rede. Por fim, advoga-se que o modelo proposto neste trabalho é escalável, pois permite uma estimativa confiável do desempenho da aplicação com uma fração da intrusão de rede necessária para realizar medição utilizando tráfego real de aplicação.

3. LEAP: Um Modelo para Predição Integrada de AppQoS e QoE

Para permitir as predições recém referidas na seção anterior, propõe-se o modelo LEAP (*Lightweight AppQoS and QoE Predictor*). O modelo objetiva fornecer ao provedor uma visão analítica sobre como o desempenho de sua rede afeta o desempenho da aplicação de *streaming* de vídeo e a respectiva experiência do usuário. A Figura 1 ilustra o esquema geral do modelo proposto. Sua construção depende de indicadores de rede e de aplicação. Como indicadores de rede, neste trabalho considera-se: (*i*) atraso, (*ii*) variação do atraso, (*iii*) vazão e (*iv*) perda. Já para aferir a qualidade da reprodução de vídeos, o presente trabalho considera: (*i*) tempo de inicialização, (*ii*) número de paradas e (*iii*) somatório do tempo das paradas. Para cada indicador de AppQoS são utilizados os quatro indicadores de QoS, construindo-se cada um dos três preditores de forma individual. Em um segundo momento, os três indicadores de AppQoS são utilizados para estimar QoE (MOS). Além do uso para predição de AppQoS e QoE para o serviço de *streaming*, defende-se que esse arcabouço de predição é flexível e suficientemente geral para ser utilizado em outras aplicações.

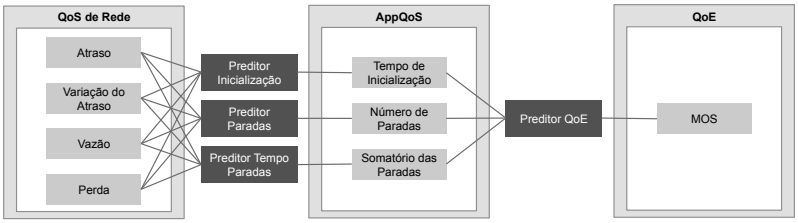


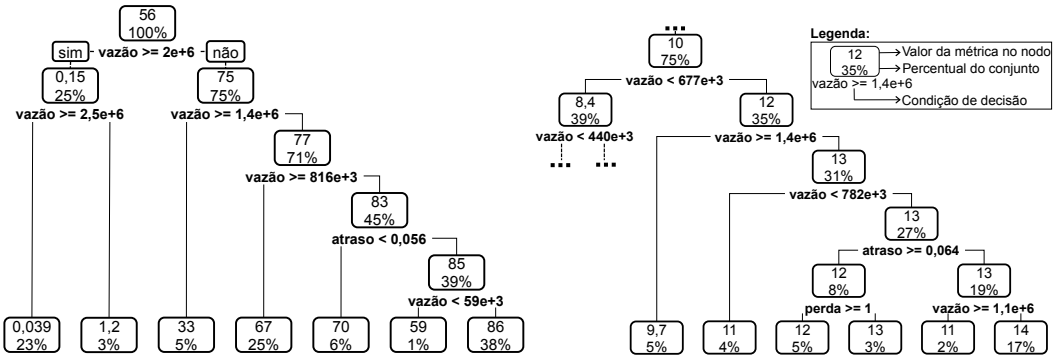
Figura 1. Esquema geral do modelo proposto

3.1. Predição de AppQoS

Neste trabalho emprega-se o modelo de predição baseado em Árvore de Decisão de Regressão em razão da capacidade desse modelo em lidar com atributos que possuem uma relação complexa e não linear entre si. Durante a fase de estudo de candidatos também foram considerados os seguintes modelos de aprendizado de máquina: Regressão Linear Múltipla, Árvores de Decisão, Redes Neurais Artificiais, Gaussian Naive Bayes e Suport Vector Regression. Esses, contudo, incorreram em desempenho inferior, em termos de acurácia, quando da execução de processo de aprendizagem via protocolo NCV (*Nested Cross Validation*) [Tsamardinos et al. 2014].

Durante a fase de treinamento, o modelo aprende a influência de cada atributo X na variável de resposta Y. Nesse caso, os indicadores de QoS são os atributos (X) e os indicadores de desempenho da aplicação de vídeo são as variáveis de resposta (Y). Após construídas as árvores, para qualquer nova instância de valores de QoS, é possível realizar a predição de cada indicador de AppQoS apenas comparando, em cada árvore, os valores X com os limiares estabelecidos, nó após nó, até a chegada em um nó final, onde estará a

predição do modelo. Por exemplo, na Figura 2 é possível observar segmentos das árvores de decisão que realizam a predição para os indicadores AppQoS de número de paradas e duração de paradas para a resolução 1080p. Os aspectos construtivos e a avaliação da acurácia dessas árvores serão explicadas na Subseção 4.3.



(a) Duração das Paradas (b) Número de Paradas

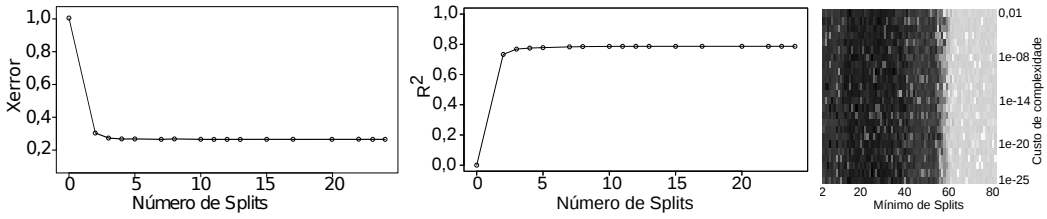
Figura 2. Árvores de decisão para duração e número de paradas para 1080p

Para construir cada árvore de decisão é empregado um procedimento recursivo que vai construindo uma árvore binária por meio de sucessivas divisões, partindo de um nó único que contém todos os atributos *X*. Esse procedimento ocorre balizado por dois parâmetros que controlam o crescimento da árvore: um limiar *minsplit*, que representa o número mínimo de observações associadas para que um novo ramo seja criado, e um segundo parâmetro *cp*, que indica o ganho mínimo na redução do erro que um novo ramo precisa proporcionar para ser criado. Como saída do processo de geração, cada árvore possui associada a si um erro mínimo de validação cruzada *xerror*, que consiste em uma estimativa do erro que será verificado ao se submeter dados independentes (ou seja, não utilizados no treinamento) à estrutura criada. Em um segundo momento, realiza-se o processo de poda. Nesse procedimento, para cada par de nós finais, com um pai comum, avalia-se o erro médio quadrático (MSE) quando da remoção dos nós finais, conforme a Equação 1, fazendo do nó pai um nó final. Se o erro diminui, a árvore é podada, senão é mantida igual para esse nó.

$$MSE = \sum_{c \in leaves(T)} \sum_{i \in c} (y_i - m_c)^2 \quad (1) \qquad Qini = -0,963 \times \log_{10}(t_0 + 5,381) + 5 \quad (2)$$

Como um exemplo da aplicação desses parâmetros construtivos, a Figura 3 ilustra três aspectos importantes para a construção do preditor de tempo de inicialização para vídeos 1080p. Em (a) e (b) é possível verificar que tanto o erro de validação cruzada quanto o R^2 estabilizam depois de 5 *splits*. Pode-se afirmar, então, que deixar a árvore crescer para mais de 5 bifurcações não ajudaria a diminuir o erro ou melhorar o ajuste do preditor. Já em (c) é possível observar o mapa de calor que ilustra a matriz gerada pela interpolação dos parâmetros *minsplit* e *cp*, onde as zonas mais escuras representam valores menores de *xerror*. Objetivamente, para o preditor em questão chegou-se a um *minsplit*=24, *cp*=10^{-22} e número de *splits*=5, que representam a parametrização ótima do modelo.

Antes de encerrar a apresentação do modelo, é importante destacar que seu modo de operação pode ser dividido em duas etapas. A primeira etapa, que ocorre de maneira



(a) Xerror VS Split (b) R^2 VS Split (c) Mapa de Calor

Figura 3. Aspectos construtivos do preditor de tempo de inicialização para 1080p

offline, consiste na seleção e no treinamento do modelo. Nessa etapa são executadas tarefas de alta complexidade. Mais especificamente, a geração das Árvores de Decisão binárias ótimas constitui um problema NP-Completo, exigindo o emprego de algoritmos heurísticos para obtenção de uma árvore *near-optimal*. Já a aplicação do modelo de predição é feita de forma *online*, ou seja, na medida em que os dados de QoS estão disponíveis, os indicadores de AppQoS e QoE são calculados e disponibilizados.

3.2. Predição de QoE

Complementando o trabalho do presente artigo, o LEAP permite utilizar os indicadores de desempenho de aplicação para estimar a qualidade da experiência do usuário. Para estabelecer a relação entre os indicadores de AppQoS e a nota MOS atribuída por um usuário o modelo lança mão de uma adaptação da formulação matemática oriunda de recentes trabalhos de pesquisa que, por meio de testes com usuários de serviços de *streaming* de vídeo, relacionaram a influência do tempo de inicialização de vídeos com a degradação da qualidade de experiência do usuário [Seufert et al. 2014] [Hoßfeld et al. 2012]. A influência do tempo de inicialização (t_0) na nota MOS pode ser definida pela Equação 2, onde *Qini* é a nota MOS influenciada e t_0 é o valor predito para o tempo de inicialização. Os valores 0,963 e 5,381 foram calculados através da resolução de um problema de minimização não-linear do erro médio quadrático entre os valores de MOS em t_0 e a função $f(t_0)$.

$$\lambda = \begin{cases} \frac{\sigma}{\sigma+\rho}, & \text{se } \sigma + \rho < T \\ \frac{\sigma}{T}, & \text{caso contrário} \end{cases} \quad (3) \qquad Qst = \begin{cases} 1, & \text{se } n > 6 \\ a_i \times e^{-b_i \times n} + c_i, & \text{se } n \leq 6 \end{cases} \quad (4)$$

Já para modelar a influência das interrupções na percepção da qualidade de exibição de um vídeo, primeiramente é necessário definir, conforme a Equação 3, um fator λ que calcula a razão entre o tempo total σ em que o vídeo ficou parado e o intervalo transcorrido a partir do início da observação, dado pela soma de σ com o tempo efetivo ρ de exibição do vídeo. De acordo com [Casas et al. 2013], essa observação é realizada dentro de *slots* T com duração típica de um minuto, o que permite generalizar o método para vídeos de qualquer duração. Uma vez que se tenha calculado λ, observa-se seu valor para definir um nível $1 \leq i \leq 5$ de degradação, dado pelos intervalos de λ descritos nas colunas da Tabela 1. O MOS *Qst* é, então, calculado de acordo com a Equação 4, onde a_i, b_i, c_i são constantes definidas $\forall i = 1, 2, 3, 4, 5$ de acordo com a Tabela 1, e n é o número de paradas dentro do *slot* T. Pode-se observar na referida equação que para um número de paradas maior que seis, a qualidade estará totalmente comprometida, o que resulta em um MOS com nota 1.

Tabela 1. Tabela de fatores em razão de λ para cálculo de MOS *Qst*

Fator	$\lambda < 0,05$	$0,05 \leq \lambda < 0,10$	$0,10 \leq \lambda < 0,20$	$0,20 \leq \lambda < 0,50$	$\lambda \geq 0,50$
a	3,012682983	3,098391523	3,190341904	3,248113258	3,302343627
b	0,765328992	0,994413063	1,520322299	1,693893480	1,888050118
c	1,991000000	1,901000000	1,810138616	1,751982415	1,697472392

4. Construção do Modelo

Esta seção apresenta os aspectos práticos relacionados com a construção do modelo de predição. Primeiramente, aborda-se a aquisição de indicadores de QoS e AppQoS. Logo após, descreve-se detalhes do ambiente de treinamento do modelo e, por fim, apresenta-se e discute-se os modelos de predição resultantes.

4.1. Aquisição de Indicadores de QoS e AppQoS

Para obtenção de indicadores de desempenho de rede, empregou-se a plataforma NetMetric [dos Santos et al. 2007]. A plataforma NetMetric teve sua primeira implementação funcional em 2007 e, desde então, está em constante desenvolvimento para adaptar-se aos padrões de medição em redes IP, principalmente os definidos pelo IETF IPPM. Para tanto, emprega técnicas de medição ativa, em que rajadas controladas de pacotes são injetadas em um ponto de origem em direção a um ponto de destino, permitindo avaliar a deformação dos pacotes ao longo da rede e extrair indicadores de desempenho da rede IP.

O NetMetric possui uma entidade Gerente, que permite configurar agendas de testes a serem executados por entidades Agente. Por sua vez, uma entidade Agente pode ser utilizada tanto como origem de uma medição (fonte) quanto como destino (refletora). A plataforma foi configurada para trabalhar com conjuntos de duas rajadas de pacotes distintas. Uma primeira rajada utiliza o protocolo UDP e serve para medir atraso de uma via (OWD - *One Way Delay*), variação do atraso (*jitter*), bem como perda de pacotes, também em um sentido. Cada instância dessa rajada injeta um conjunto de 400 pacotes de 100 bytes separados por um *gap* de 50 ms. Já para medir vazão, é utilizada uma rajada com o protocolo TCP contendo 640 pacotes de 1.488 bytes para cada sondagem entre dois pontos. As duas rajadas somadas geram um volume unidirecional de 992 KB. Embora a plataforma NetMetric seja capaz de realizar medições de forma bi-direcional, optou-se por extrair indicadores de forma unidirecional a fim de isolar o canal foco desse estudo, ou seja, o *downlink* de redes LTE.

Para obtenção dos indicadores de AppQoS foi desenvolvido um módulo para a plataforma NetMetric, materializado na forma de um *plugin* para o navegador Chrome, que viabiliza a extração de indicadores de desempenho relacionados à reprodução de vídeos diretamente do controlador de vídeo nativo do HTML5. A justificativa para tal escolha reside na migração de grandes provedores de *streaming* de vídeo para a tecnologia HTML5, tal como o Youtube e o Netflix.

Para viabilizar a medição da camada de aplicação, o NetMetric foi configurado para que o agente fonte realize reproduções periódicas de vídeos via navegador Chrome, que recupera vídeos armazenados no agente refletor. Para realização do treinamento do algoritmo foram utilizados dois vídeos de sessenta segundos cada, todos no formato MPEG v4 e codificados em H.264, com resoluções de 720p e 1080p, com volumes de 9,2 MB e 14,3 MB respectivamente. Cabe destacar que a transferência de vídeos em diferentes resoluções torna-se necessária, pois cada resolução possui um *bitrate* diferente e, dessa forma, exige um nível diferenciado de desempenho da camada de rede. Nesse sentido, cada um dos três preditores de parâmetros de AppQoS necessita ser treinado de forma

isolada para cada resolução de vídeo, permitindo que o algoritmo de treinamento aprenda a demanda de rede para reprodução de vídeos em cada resolução.

4.2. Ambiente de Treinamento do Modelo

Uma abordagem possível para obtenção do conjunto de dados que relaciona QoS e AppQoS seria a realização das medições em uma rede LTE em produção, realizando as medições da camada de rede e de aplicação em paralelo, objetivando garantir que as condições de rede não seriam alteradas durante as medições nas duas camadas. No entanto, em razão da natureza invasiva das duas estratégias de medição, a medição concomitante nas duas camadas implicaria em uma competição por recursos compartilhados de rede, resultando em uma interferência mútua entre as técnicas. Uma outra abordagem possível poderia prever a serialização das medições no tempo. No entanto, devido à grande variabilidade de desempenho em redes LTE, observada mesmo em pequenos intervalos de tempo, concluiu-se que esse não seria um ambiente adequado para treinar o modelo. O treinamento exige, obrigatoriamente, que a condição de rede permaneça estática entre a aferição de desempenho da rede e da aplicação. Caso contrário, o algoritmo de aprendizado estabeleceria relações erradas sobre a influência de cada parâmetro de rede nas variáveis de resposta.

Diante dessas reconhecidas dificuldades, optou-se por criar um ambiente controlado que emula de forma fiel as condições de rede observadas em um ambiente de produção. Para emular as condições de vazão, atraso, variação de atraso e perda de pacotes inerentes aos ambientes LTE, foi empregada a solução WANEM [Kalitay and Nambiar 2011]. A ferramenta WANEM permite impor, de forma controlada, condições variadas de rede ao tráfego de interesse. A Figura 4 ilustra a topologia utilizada para implantar o ambiente controlado.

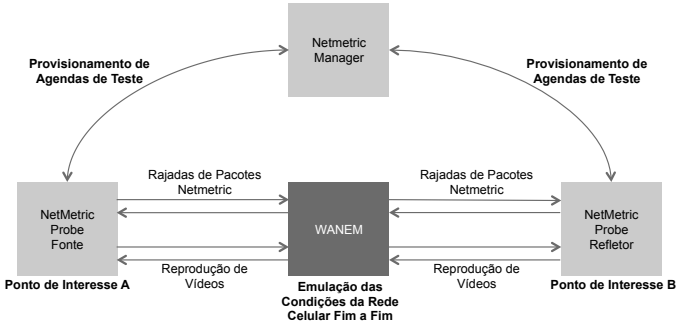


Figura 4. Ambiente controlado de treinamento

A escolha dos valores utilizados para a configuração do WANEM foi feita com base em 7.450 medições realizadas pelo NetMetric em uma rede LTE de abrangência nacional em produção[3], entre maio e outubro de 2015. A Tabela 2 enumera os quatro níveis selecionados para cada indicador de QoS. Para convergir para os referidos valores, cada indicador foi testado individualmente quanto à normalidade de suas distribuições com um teste de Shapiro-Wilk, a uma signifiância $\alpha = 0,05$. Em razão da distribuição normal, os indicadores de atraso, variação do atraso e vazão foram segmentados por meio da análise de quartis. Já os valores para o indicador de perda foram selecionados via análise de moda para valores inteiros de perda.

[3]Por questões de confidencialidade, não é possível apresentar caracterização detalhada da rede.

Por meio da realização de experimentos preliminares (via *design Full Factorial* $2^{k.r}$), constatou-se que o indicador de variação de atraso não ajudava a explicar o modelo de regressão proposto na seção anterior. Uma vez determinados os parâmetros relevantes (vazão, atraso e perda), foi realizado um segundo conjunto de experimentos, dessa vez com o *design Full Factorial* $4^{k.r}$, permitindo uma maior variação de níveis para cada parâmetro sem incorrer em um número exagerado de experimentos. Com três parâmetros de entrada e quatro níveis, o *design* resultou em 64 combinações possíveis. Por meio da observação da variabilidade dos resultados foi definido o número de dez replicações para cada combinação, considerando-se um nível de significância de 95%.

Tabela 2. Valores da rede LTE real usados no treinamento do modelo

Indicador	Valor
Vazão	0,9 Mbps; 8,8 Mbps; 15,5 Mbps; 25,3 Mbps
Atraso	22 ms; 56 ms; 64 ms; 98 ms
Perda	0%; 2%; 5%; 13%

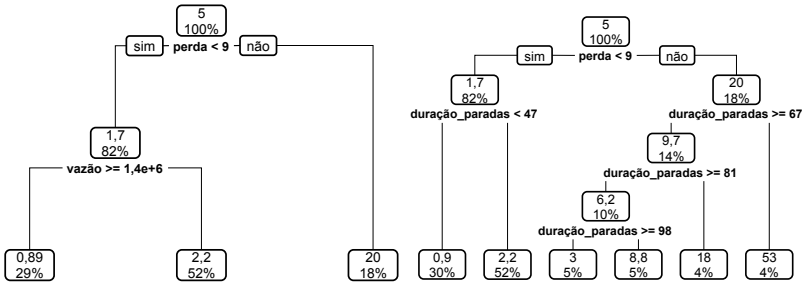
4.3. Preditores Resultantes

Nesta etapa, com os dados medidos no ambiente controlado, trabalhou-se na geração de árvores de decisão binária para cada um dos três indicadores de AppQoS preditos. Essas árvores devem ser específicas para cada uma das duas resoluções testadas (720p e 1080p), totalizando a derivação de seis dessas estruturas.

As árvores das Figuras 2 e 5, assim como as demais geradas no escopo deste trabalho, refletem as podas ótimas das estruturas. Na Figura 2(a) está representada a árvore para o tempo total de parada de um vídeo, na resolução 1080p, em função dos três indicadores primários de rede. O nodo raiz, que simboliza o primeiro critério de decisão da árvore, é dado em função da vazão de dados observada na rede (representada em *bits* por segundo). O valor indicado neste nodo, 2 Mbps, está relacionado ao *bitrate* do vídeo utilizado nos testes - em torno de 1,9 Mbps. Para valores de vazão acima dessa taxa, a duração média do total de paradas (0,15 segundos, de acordo com o valor central indicado pelo nodo filho à esquerda da raiz) é consideravelmente menor do que a observada para redes com uma taxa menor (75 segundos, conforme registrado pelo nodo imediatamente à direita da raiz). Isto indica que o fato de a rede ter capacidade para dar suporte ao *bitrate* do vídeo é o fator mais importante para determinar o comportamento macro de sua exibição.

Outro aspecto relacionado é a existência de uma faixa intermediária acima do *bitrate* em que há uma pequena degradação nos indicadores de exibição do vídeo - no caso da Figura 2(a), entre 2 Mbps e 2,5 Mbps. Nessa faixa, qualquer variação instantânea na vazão pode levar a uma necessidade de preenchimento de *buffer*, levando à observação dessa degradação. Para valores acima de 2,5 Mbps, no entanto, a reprodução do vídeo se dá praticamente sem interrupções, independente do valor de qualquer outro indicador preditor. Destaca-se que um comportamento semelhante foi observado para 720p, em que o nodo raiz foi determinado por uma vazão idêntica ao *bitrate* do vídeo (1,2 Mbps), mas a reprodução sem falhas foi mapeada apenas para vazões superiores a 1,9 Mbps.

Para o indicador de quantidade total de paradas na resolução 1080p, o nodo raiz, assim como a porção da árvore à sua esquerda, são análogos aos seus correspondentes para o indicador de duração total de paradas. Isto significa que a primeira decisão é tomada em relação a uma vazão semelhante ao *bitrate*, e que valores superiores a esse parâmetro ainda apresentam degradações até um segundo limiar de vazão (2,3 Mbps). No entanto, o ramo destacado pela Figura 2(b), que representa a porção dessa árvore conectada imediatamente à direita do nodo raiz, tem um comportamento distinto dos observados até então.



(a) Tempo de Inicialização

(b) Tempo de Inicialização, levando em conta o tempo total de parada

Figura 5. Árvores de decisão para tempo de inicialização na resolução 1080p

Para esse subconjunto, valores entre 677 kbps e 2 Mbps (limitação dada pelo nodo raiz) possuem em média 12 paradas, uma quantidade maior se comparada àquela observada para vazões menores (8,4 paradas em média). Uma análise conjunta com o tempo total de paradas no conjunto de treinamento mostra que, para as amostras em que a velocidade fica abaixo de 677 kbps, a média do tempo total em que o vídeo fica interrompido é de 83,76 segundos. No caso complementar, essa mesma grandeza é avaliada em 64,98 segundos. Pode-se concluir, portanto, que o vídeo efetivamente fica mais tempo parado em redes mais lentas, mas que o intervalo demandado por cada interrupção é menor.

Para o tempo de inicialização de vídeo, de acordo com a Figura 5(a), a decisão no nodo raiz é tomada por uma condição muito degradada de rede (perda de pacotes acima de 9%). De acordo com a Tabela 3, os fatores de ajuste deste preditor em relação aos valores efetivamente medidos foram baixos, com $R^2 = 0,3147$ no melhor caso, para a resolução 1080p. Isto indica a insuficiência dos três indicadores de QoS de rede para a derivação do indicador de tempo de inicialização. Para tornar a predição mais precisa, buscou-se, entre todos os indicadores envolvidos, aqueles em que a correlação de Pearson com o indicador em questão fosse mais forte. Verificou-se que, além dos indicadores já utilizados na árvore (perda e vazão), o próximo indicador com maior correlação era o de duração total de paradas, cujo vínculo com o tempo de inicialização mostrou-se cinco vezes mais forte do que o existente entre esse indicador e o atraso unidirecional, por exemplo. A Figura 5(b) ilustra a árvore de decisão gerada com a utilização do tempo total de paradas como variável de entrada. A predição resultante da submissão das amostras de teste a essa estrutura levou a um crescimento do fator de ajuste, que subiu para $R^2 = 0,8085$ para o vídeo 1080p, com redução também na raiz do erro quadrático, conforme pode ser observado na coluna Tempo de Inicialização 2 da Tabela 3.

Tabela 3. RMSE e R^2 dos preditores para 720p e 1080p

	Duração Paradas		Total Paradas		Tempo Inicialização		Tempo Inicialização 2	
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2
720p	5,7548 s	0,9627	1,4747	0,8222	4,2321 s	0,2169	2,0151 s	0,8774
1080p	9,8849 s	0,9274	1,1847	0,9449	6,5461 s	0,3147	3,9102 s	0,8085

5. Avaliação de Desempenho

Esta seção apresenta a avaliação de desempenho do modelo proposto. Adicionalmente, a seção aborda os resultados potenciais da aplicação do modelo na rede da operadora móvel mencionada ao final da Subseção 4.2. Essencialmente, busca-se responder as seguintes perguntas de pesquisa: (*i*) Que nível de acurácia é possível obter-se ao predizer o desempenho da aplicação de vídeo com base na observação do desempenho de rede? (*ii*) Qual

o ganho, em termos de redução de intrusividade, que o modelo proposto fornece quando de sua aplicação em uma rede LTE de grande porte? (*iii*) De que maneira um provedor poderia tirar proveito de uma predição integrada de qualidade de vídeo e QoE?

5.1. Acurácia do Modelo

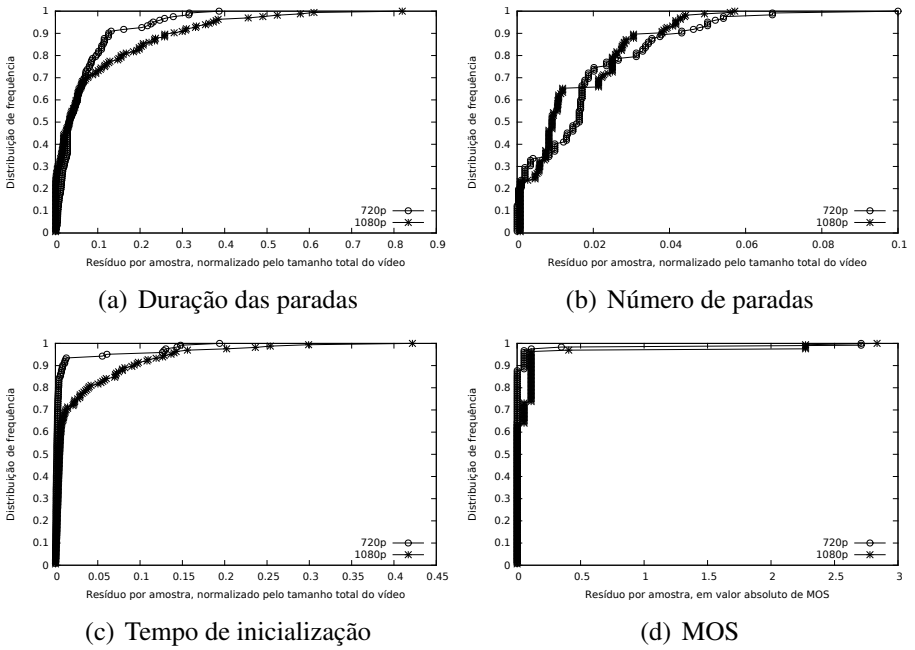
A acurácia do modelo proposto pode ser avaliada com a submissão do conjunto de teste, o qual é independente dos dados de treinamento, para predição das variáveis representadas. Cada amostra i deste conjunto contém um valor medido para as três variáveis preditoras (vazão, atraso e perda) e para os três indicadores de aplicação (duração de paradas, contagem de paradas e tempo de inicialização de vídeo). Utilizando-se o primeiro grupo de variáveis como entrada das árvores representadas nas Figuras 2 e 5, obtém-se uma saída estimada para cada um dos indicadores do segundo grupo, que então passam a ter um valor observado x_i e um valor predito $\hat{x}_i$. Isto permite calcular, para cada amostra i, o resíduo normalizado r_i, definido pela Equação $r_i = |\hat{x}_i - x_i|/N$. O fator N é utilizado para normalização dos erros. Para os três indicadores de aplicação, o valor de N é derivado da duração dos vídeos utilizados nos experimentos realizados no ambiente controlado (60 segundos), o que permite generalizar a avaliação do método para um vídeo de extensão arbitrária.

Os gráficos da Figura 6 variam r_i no eixo horizontal, associando cada valor desse eixo a uma porção do conjunto amostral (no eixo vertical) em que o próprio valor de r_i é igual ou inferior ao limiar fixado. Assim, considerando a resolução 1080p na Figura 6(c), o valor 0,093 no eixo horizontal está associado ao valor 0,9 no eixo vertical, indicando que 90% das amostras do respectivo conjunto de teste apresentam $r_i \leq 0,093$. Já para a resolução 720p, no mesmo gráfico, 90% das amostras possuem $r_i \leq 0,0086$. Isso pode ser interpretado como um erro de 9,3% e 0,86%, respectivamente. Calculando-se a média dos valores de r_i associados a 90% das amostras para todos os indicadores, nas duas resoluções, tem-se um valor médio de $\bar{r_i} = 0,0982$ (9,8%).

Também com base nos valores observados e preditos dos indicadores de aplicação, é possível calcular o MOS de acordo com a abordagem apresentada na Subseção 3.2. Para esse indicador, que é independente da duração do vídeo, foi usado um valor $N = 1$, ou seja, os erros apresentados são dados em função da escala absoluta de MOS, que varia de 1 a 5. De acordo com a Figura 6(d), a predição de MOS apresentou um erro de até $r_i = 0,11$ para 90% dos casos do conjunto de teste 1080p. Analisando o significado desse indicador com um exemplo, tem-se que, se um $MOS = 3,5$ for calculado utilizando os indicadores de aplicação observados, um valor $3,39 \leq \widehat{MOS} \leq 3,61$ será computado com a predição desse indicadores em 90% das situações. Para a resolução 720p, o erro foi ainda menor, apresentando o valor $r_i = 0,05$. Embora o erro dos preditores de AppQoS já seja bem baixo, o impacto desse erro é ainda menor quando se usa os indicadores de AppQoS para predição de QoE, conforme ilustrado na Figura 6(d).

5.2. Intrusividade do Modelo

A Figura 7(a) ilustra um comparativo de intrusividade, montado com parâmetros de volume descritos na Subseção 4.1, entre o modelo proposto e a medição de qualidade por meio da transferência de vídeos reais. Como é possível observar, o caso mais intrusivo de LEAP (intervalo de *polling* de 10 minutos) para 500 sondas, o volume de dados gerados atinge 66,65 GB/dia. Para a mesma configuração, a estratégia da transferência de vídeos reais em série para medição da qualidade da rede geraria um volume de 1.574,71 GB/dia. Ou seja, a intrusividade das técnicas convencionais é mais de vinte vezes maior do que a observada ao se empregar LEAP.



(a) Duração das paradas (b) Número de paradas

(c) Tempo de inicialização (d) MOS

Figura 6. CDFs dos erros para predição de AppQoS e QoE em vídeos de 1080p

Complementando a análise recém apresentada, destaca-se, ainda, que a alta variabilidade de desempenho da rede LTE, justificada principalmente pela mobilidade dos usuários e pela alta competição por recursos da interface de rádio, exige uma frequência elevada de monitoramento de qualidade para que o operador de serviços tenha uma visão realista do ambiente de rede. Nesse cenário, o aspecto de baixa intrusividade é indispensável para que uma solução de monitoramento de desempenho seja escalável. Caso contrário, o tráfego gerado para medição da qualidade da rede consumiria um grande volume de recursos, gerando uma diminuição significativa dos recursos disponíveis para os usuários da rede.

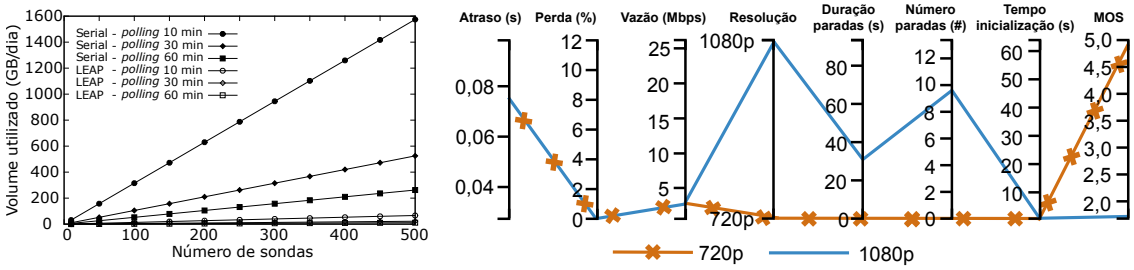
5.3. Aplicação do Modelo

Após construído, o modelo foi colocado em operação em uma operadora celular de grande porte, já referida na Subseção 4.2. Para tanto, foi definido um ponto de interesse para instalação do agente refletor, que foi configurado junto ao servidor CDN da operadora. De forma complementar, foram instalados agentes fonte espalhados por todo o território nacional. Essa configuração permitiu o monitoramento sistemático da qualidade do serviço de vídeo em um contexto fim-a-fim, entre o ponto de interesse refletor e os pontos de interesse regionais.

Por restrição de espaço, apresenta-se agora um único exemplo concreto de uso do modelo. Para apresentar os resultados consolidados foi utilizado um gráfico de coordenadas paralelas que evidencia o relacionamento dos diferentes indicadores em uma mesma sessão de medição. Na Figura 7(b) é possível observar duas linhas em destaque. As duas linhas compartilham uma condição idêntica de rede: 77 ms de atraso unidirecional, 0% de perda de pacotes e 1,950 Mbps de vazão. Para a mesma condição de rede, foi utilizado o preditor para inferir o comportamento da aplicação. Para o vídeo de 720p o número de paradas foi zero, o tempo total de paradas também foi zero e, consequentemente, o MOS foi próximo de cinco. Já para o vídeo de 1080p, para as mesmas condições de rede, o número de paradas foi 9,68, o tempo total de paradas foi 32,9 segundos e o tempo de

inicialização de vídeo foi 0,89 segundos. Esse comportamento de aplicação levou a uma nota MOS 1,75.

A predição integrada permite que o provedor de serviços utilize o indicador de QoE para isolar rapidamente qualquer degradação significativa para o serviço de *streaming* de vídeo na rede LTE. Em um segundo momento, os indicadores de AppQoS permitem que o operador da rede identifique qual é o principal ofensor da degradação de QoE (no caso analisado, número e duração de paradas). Isto feito, basta que o operador observe o indicador de QoS que mais influencia o comportamento do indicador de AppQoS de interesse, por meio da observação da Árvore de Decisão relacionada (no caso, como indicadores mais influentes destaca-se vazão seguido por atraso unidirecional). Fechado esse ciclo, o operador pode realizar intervenções na rede (por exemplo, eliminação de gargalos para aumento de vazão e/ou ajustes de filas de prioridade para diminuição de atraso), para logo após observar seus reflexos ao longo das camadas de rede, aplicação e experiência do usuário.



(a) Intrusividade

(b) Visualização *online* de indicadores

Figura 7. Análise de intrusividade e caso de uso do modelo

6. Conclusão e Trabalhos Futuros

Os experimentos realizados evidenciaram a viabilidade de realizar a estimativa de qualidade da aplicação de *streaming* de vídeo utilizando indicadores de QoS da rede como preditores. A estimativa apresentou um erro médio para AppQoS menor que 9,92%, e um erro para a nota MOS menor que 0,11 pontos para mais de 90% dos casos. Além de acurado, o modelo proposto gera um volume de dados que representa, em média, apenas 4% da intrusividade de técnicas convencionais que necessitam transferir vídeos reais para medição de desempenho. Essa baixa intrusividade permite ao provedor realizar uma medição sistemática e com um pequeno intervalo entre medições, sem que isso consuma um volume excessivo de recursos da rede. É importante mencionar que para garantir que os preditores não percam acurácia ao longo do tempo, principalmente em razão da evolução tecnológica ou da alteração de configurações da rede LTE, o modelo precisa passar por um processo de recalibração periódica. Por ser executado de forma *offline*, e durar na ordem de poucas horas, pode ser conduzido com a frequência desejada pelo operador.

O modelo proposto mostra-se efetivo ao fornecer a operadores de rede informações para suporte à decisão, principalmente durante a execução de rotinas de configuração e detecção de degradações. Para prover subsídios sobre como cada resolução de vídeo irá desempenhar sob determinada condição de rede, optou-se por trabalhar com vídeos de resolução fixa. Como trabalho futuro, pretende-se evoluir o modelo para prover uma predição complementar considerando mecanismos adaptativos, tal como DASH. Além disso, como trabalho futuro vislumbra-se a aplicação do modelo para outros serviços, tais como aplicações de VoIP e IPTV, objetivando ratificar a generalidade do modelo. Por

fim, pretende-se explorar melhor os modelos de QoE por meio da condução de testes com usuários reais para validar ou aprimorar os recentes modelos que realizam o mapeamento AppQoS para QoE.

Referências

Balachandran, A., Sekar, V., Akella, A., Seshan, S., Stoica, I., and Zhang, H. (2012). A quest for an internet video quality-of-experience metric. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 97–102.

Balachandran, A., Sekar, V., Akella, A., Seshan, S., Stoica, I., and Zhang, H. (2013). Developing a predictive model of quality of experience for internet video. In *Proceedings of the ACM SIGCOMM 2013*, pages 339–350.

Casas, P., Schatz, R., and Hoßfeld, T. (2013). Monitoring youtube qoe: Is your mobile network delivering the right experience to your customers? In *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, pages 1609–1614.

De Pessemier, T., De Moor, K., Joseph, W., De Marez, L., and Martens, L. (2013). Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching. *IEEE Transactions on Broadcasting*, 59(1):47–61.

dos Santos, G. L., Guimaraes, V. T., Silveira, J. G., Vieira, A. T., de Oliveira Neto, J. A., da Costa, R., and Balbinot, R. (2007). Uama: a unified architecture for active measurements in ip networks; end-to-end objetive quality indicators. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 246–253.

Hoßfeld, T., Egger, S., Schatz, R., Fiedler, M., Masuch, K., and Lorentzen, C. (2012). Initial delay vs. interruptions: between the devil and the deep blue sea. In *Proceedings of the 4th International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 1–6.

Hsu, W.-H. and Lo, C.-H. (2014). Qos/qoe mapping and adjustment model in the cloud-based multimedia infrastructure. *IEEE Systems Journal*, 8(1):247–255.

ITU (2015). Measuring the information society. Technical report. http://www.itu.int/en/ITU-D/Statistics/Pages/publications/mis2015.aspx.

Kalitay, H. K. and Nambiar, M. K. (2011). Designing wanem: A wide area network emulator tool. In *Proceedings of the 3rd International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–4.

Katsarakis, M., Fortetsanakis, G., Charonyktakis, P., Kostopoulos, A., and Papadopouli, M. (2014). On user-centric tools for qoe-based recommendation and real-time analysis of large-scale markets. *IEEE Communications Magazine*, 52(9):37–43.

Msakni, H. and Youssef, H. (2013). Is qoe estimation based on qos parameters sufficient for video quality assessment? In *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 538–544.

Quadros, C., Santos, A., Gerla, M., and Cerqueira, E. (2015). A qoe-aware mechanism to improve the dissemination of live videos over vanets. In *Proceedings of the XXXIII Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, pages 31–40.

Seufert, M., Egger, S., Slanina, M., Zinner, T., Hobfeld, T., and Tran-Gia, P. (2014). A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492.

Tsamardinos, I., Rakhshani, A., and Lagani, V. (2014). Performance-estimation properties of cross-validation-based protocols with simultaneous hyper-parameter optimization. In *Artificial Intelligence: Methods and Applications*, pages 1–14.

Caracterização do Comportamento dos Clientes de Vídeo ao Vivo Durante um Evento de Larga Escala

Breno Santos[1] **Gustavo Carnivali**[1] **Wagner Almeida**[1,3] **Alex B. Vieira**[1]
Ítalo Cunha[2] **Jussara Almeida**[2]

[1]Departamento de Ciência da Computação, Universidade Federal de Juiz de Fora
[2]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
[3]Instituto Federal do Sudeste de Minas Gerais

{wagner.almeida, breno.santos, gustavocarnivali}@ice.ufjf.br {alex.borges}@ufjf.edu.br
{cunha, jussara}@dcc.ufmg.br

Abstract. *Internet large-scale events are no longer an exception. Although, both researchers and service providers still have limited visibility and understanding about these events. In this paper, we present an in-depth live streaming client behavior characterization during a large scale event. We analyze 64 soccer matches data from FIFA's 2014 Soccer World Cup collected at servers from one of the major Internet content providers in Brazil. We have identified the main client behavior features and propose a simple model that captures its behavior. Our results are compared to the state-of-the-art and indicate a change in the client behavior during Internet large-scale events. We believe that our results can be applied to generate realistic synthetic workloads and serve as a substrate for the development and evaluation of new Internet live streaming architectures.*

Resumo. *Eventos de larga escala na Internet não mais são exceções. Entretanto, tanto pesquisadores quanto provedores de serviço ainda têm visão e entendimento limitados sobre tais eventos. Neste artigo, nós caracterizamos o comportamento dos usuários de um sistema de vídeo ao vivo durante um grande evento na Internet, a copa do mundo de futebol da FIFA em 2014. Analisamos dados relativos à transmissão de 64 partidas por um dos principais provedores de conteúdo Internet do Brasil. Nós identificamos as principais características do comportamento de um cliente e propomos um modelo simples que captura seu comportamento típico. Nossos resultados são comparados ao estado da arte e indicam mudanças no comportamento dos clientes durante eventos de larga escala na Internet. Acreditamos que nossos resultados podem ser aplicados na geração de cargas sintéticas realistas e servir como substrato para o desenvolvimento e avaliação de novas arquiteturas de transmissão ao vivo na Internet.*

1. Introdução

Eventos em larga escala são uma realidade cada vez mais comum na Internet. De fato, desde o discurso de posse de Obama[1], provedores de serviços estão sujeitos a surtos no número de clientes. Infelizmente, os provedores de serviços são incapazes de prever com precisão a demanda desses eventos, particularmente para eventos esporádicos ou únicos, frequentemente subestimando ou superestimando a demanda. Provedores de serviço transmitem conteúdo de infraestruturas baseadas em nuvem (como centros de processamento de dados ou redes de distribuição de conteúdo) [Barroso et al. 2013]. Neste

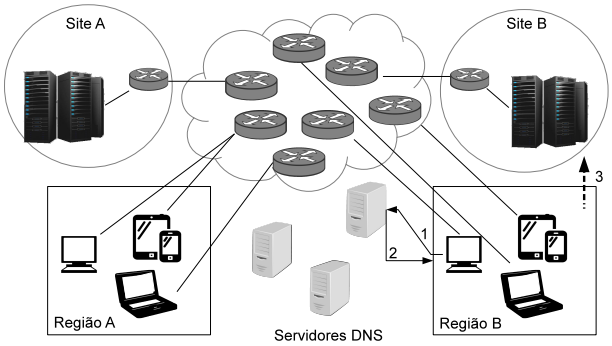


Figura 1. Arquitetura de distribuição de vídeo ao vivo do sistema estudado.

cenário, subestimar a demanda resulta em serviço de baixa qualidade e insatisfação dos clientes, enquanto superestimar a demanda implica custos adicionais de operação.

Apesar dos trabalhos de caracterização de transmissão de mídia ao vivo na existentes [Veloso et al. 2006, Hei et al. 2007, Borges et al. 2012], pesquisadores e provedores de serviço ainda têm entendimento limitado sobre eventos de larga escala na Internet. De fato, enquanto os serviços na Internet evoluem, o padrão de comportamento de seus clientes também muda [Gebert et al. 2012]. Consequentemente, indústria e academia necessitam de caracterizações e modelos atualizados, de modo que possam desenvolver aplicações mais complexas e arquiteturas de serviços mais eficientes.

Nesse cenário, nós apresentamos uma caracterização do comportamento dos usuários de um sistema de vídeo ao vivo durante a copa do mundo de futebol da FIFA em 2014 (seção 3). Nós analisamos carga de trabalho da transmissão das 64 partidas de futebol coletada em servidores de um dos principais provedores de conteúdo do Brasil (seção 2.2). Nós identificamos as principais características do comportamento de clientes e propomos um modelo simples que captura o comportamento típico de um cliente desse tipo de aplicação (seção 3.1). Nós geramos uma caracterização detalhada, permitindo assim geração de carga sintética realista. Nós contrastamos o comportamento observado durante a copa do mundo de 2014 com o comportamento observado em caracterizações de outros eventos transmitidos ao vivo pela Internet. Por último, nós desenvolvemos um gerador de carga de cargas sintéticas capaz de capturar, em momentos estáveis, o comportamento dos clientes no sistema estudado.

2. Sistema de distribuição de mídia ao vivo e conjunto de dados

2.1. Componentes do sistema

A infraestrutura de transmissão de vídeo ao vivo do provedor de serviço estudado possui dois pontos de distribuição com diversos servidores. Esses pontos estão localizados em duas das mais maiores cidades do Brasil, Rio de Janeiro e São Paulo. Como mostramos na Figura 1, em cada um desses pontos de distribuição, o provedor de conteúdo é conectado a um ponto de troca de tráfego (PTT) local e a várias redes comerciais.

O sistema de transmissão de vídeo ao vivo do provedor de serviço estudado usa *anycast* [Cesario 2012], uma técnica de engenharia de tráfego onde um prefixo IP é anun-

[1]https://gigaom.com/2009/02/07/cnn-inauguration-p2p-stream-a-success-despite-backlash/

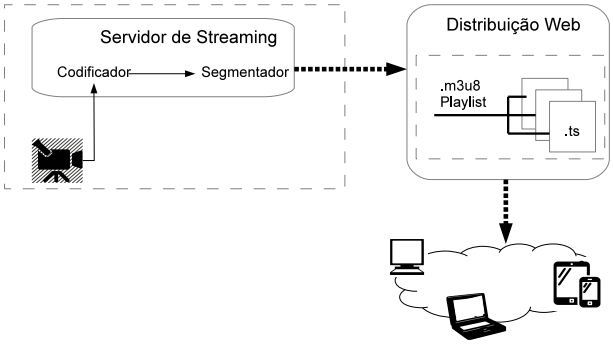


Figura 2. Esquema de codificação e distribuição do conteúdo ao vivo.

ciado a partir de múltiplos locais. A rede propaga os anúncios e protocolos de roteamento escolhem as melhores rotas. As rotas escolhidas, em geral, conectam clientes ao servidor geograficamente mais próximo, levando a menor latência, maior banda, e distribuição de carga [Katabi and Wroclawski 2000]. Assim, como ilustramos na Figura 1, um cliente acessa o domínio do provedor de conteúdo a partir de seu nome (1). O nome de domínio é resolvido por um servidor de DNS apropriado (2). O IP que o servidor de DNS resolveu corresponde a um IP anunciado via *anycast* que pode ser encaminhado a qualquer ponto de distribuição (3). Note que *anycast* divide os clientes entre os dois pontos de distribuição, mas não garante balanceamento.

Atualmente, a maioria dos servidores de envio de mídia, por exemplo, YouTube e Netflix, entregam conteúdo usando o HTTP. O HTTP é ubíquo e amplamente disponível, superando empecilhos comumente encontrados por sistemas de distribuição de mídia ao vivo que utilizando comunicação par-a-par (P2P). Em particular, transmissão por HTTP requer somente um navegador Web para visualizar conteúdos de vídeo e é desnecessário abrir portas em *firewalls* ou configurar redirecionamento de conexões externas em tradutores de endereços de rede (NATs).

O provedor de conteúdo estudado disponibiliza a mídia ao vivo em múltiplas taxas de codificação, com diferentes níveis de qualidade, usando *HTTP Live Streaming* (HLS). Um sistema com HLS funciona da seguinte maneira (Figura 2): inicialmente, o vídeo é capturado e codificado em várias taxas de transmissão (qualidade) diferentes. Cada mídia codificada é dividida em *segmentos*. Pequenos lotes de segmento com a mesma taxa de codificação são agrupados e indexados em *listas de reprodução*. A lista de reprodução permite ao cliente requisitar segmentos e reproduzir um trecho do vídeo.

Para receber o conteúdo ao vivo, cada cliente estima qual taxa de transmissão do vídeo ao vivo sua banda de rede suporta. Então, o cliente solicita a lista de reprodução referente a essa qualidade (i.e., um arquivo com extensão m3u8). Ao fim da reprodução dos segmentos de vídeo em uma lista de reprodução, o cliente novamente estima sua capacidade de rede e faz a requisição para a próxima lista de reprodução.

2.2. Conjunto de dados

Os dados analisados neste trabalho foram retirados de registros de acesso (*logs*) gerados pelos servidores de mídia ao vivo durante a transmissão dos 64 jogos da copa do mundo de futebol da FIFA, realizado entre junho e julho de 2014.

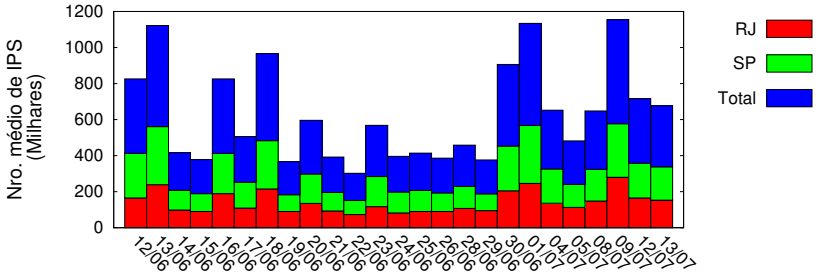


Figura 3. Média de público Internet para cada dia de transmissão da copa do mundo de futebol FIFA 2014.

Todos os jogos do evento foram transmitidos por redes de televisão, por rádio e pela Internet. Foram coletados registros de acesso da transmissão pela Internet em todos os dias em que ocorreram jogos. O conjunto de dados estudado está sumarizado na Figura 3. Em geral, os servidores atenderam entre 300 mil e 1,1 milhões de IPs únicos durante a transmissão de cada jogo, com picos de até 320 mil endereços IP únicos simultâneos. Observamos, para um único jogo, um volume total de tráfego de rede de até 300 TB e picos de até 60 GB/s.

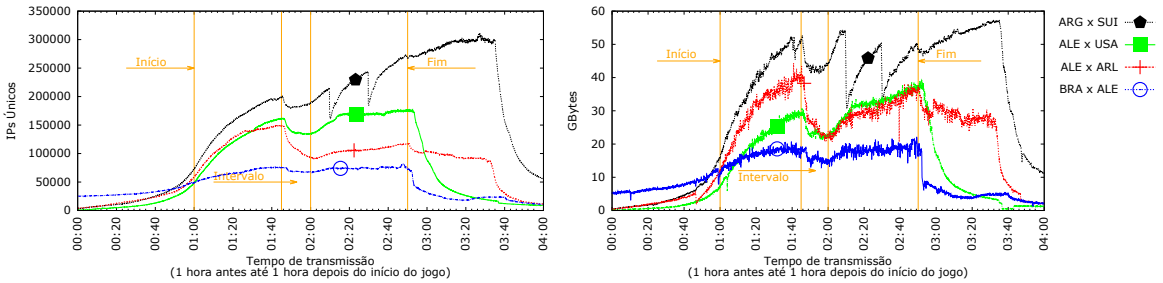
Como esperado, a carga observada durante a Copa do Mundo é maior que a carga observada durante um evento anterior: a Copa das Confederações 2013 [de Almeida Junior et al. 2015a]. O aumento de quase 4 vezes na quantidade de clientes e no volume de tráfego representou um desafio para o provedor de conteúdo naquele que foi possivelmente o evento de maior audiência no Brasil transmitido ao vivo pela Internet.

Os registros de acesso coletados dos servidores de mídia do provedor de conteúdo não incluem identificadores de sessão de usuários. Em outras palavras, apenas o endereço IP originando cada requisição é registrado. Consideramos que cada endereço IP observado corresponde a um cliente. Esta abordagem pode subestimar a quantidade de clientes em redes que utilizam NAT. Não é possível distinguir quais são as requisições de conteúdo esportivo ao vivo das requisições de mídia usuais, no entanto nossa análise não sofre impactos significativos pois elas possuem uma quantidade de tráfego desprezível.

2.3. Características gerais da carga

Diferente da copa das confederações em 2013, as configurações gerais do ambiente de transmissão de mídia ao vivo não sofreram alterações consideráveis durante os 64 jogos avaliados [de Almeida Junior et al. 2015b, de Almeida Junior et al. 2015a]. Por exemplo, durante a copa das confederações observamos alterações consideráveis das opções de qualidade de mídia sendo transmitida. Mais ainda, nossos estudos, apresentados na próxima seção, mostram que as caracterizações não revelam diferenças consideráveis no comportamento dos clientes durante uma transmissão ao vivo na copa do mundo. Por esses motivos, e por restrições de espaço, neste trabalho, focamos nossa análise em quatro jogos de interesse especial na Copa do Mundo de 2014:

- *Argentina x Suíça (Terça-feira)*: o jogo com maior carga no provedor de conteúdo;
- *Alemanha x EUA (Quinta-feira)*: o jogo mais transmitido na Internet;
- *Alemanha x Argélia (Segunda-feira)*: um jogo com alta carga e com prorrogação;
- *Alemanha x Brasil (Terça-feira)*: o fatídico jogo mais "*twittado*."



(a) Número de Usuários (b) Banda de Rede

Figura 4. Carga de trabalho a respeito dos 4 jogos de interesse.

Dentre esses jogos, apenas a partida entre Alemanha e EUA ocorre simultaneamente a outra (Portugal e Gana). A Figura 4 retrata a carga de trabalho imposta pelos quatro jogos aos servidores do provedor de conteúdo. Observe que o número de endereços IP distintos (clientes) é relativamente baixo no evento de pré-jogo, programação relativa ao jogo que se inicia cerca de uma hora antes das partidas. Entretanto, ainda no final do pré-jogo (há poucos minutos do início da partida), o número de clientes que acessa o sistema cresce de forma rápida. Geralmente, o comportamento padrão é (i) crescimento do número de clientes até minutos após o início da partida; (ii) um período razoavelmente estável durante o 1º e o 2º tempo da partida; e (iii) um intervalo caracterizado por uma queda e posterior crescimento do número de clientes. Este comportamento é estendido em jogos com prorrogação, que atraem clientes por um período de tempo mais longo. A única partida das quatro analisadas que diverge do comportamento padrão acontece entre Argentina x Suiça. Acreditamos que isso tenha ocorrido por ajuste no servidor por parte do provedor de conteúdo ao perceber o crescimento considerável no número de requisições. Nesse período há um crescimento de 100% nas requisições com erro 404.

Observamos também que o dia da semana de realização do jogo é um fator de grande impacto na atração de usuários. Esse impacto fica mais claro ainda na Figura 5, onde apresentamos a taxa de novas sessões de clientes por segundo, avaliada no período do pré-jogo e do 1º tempo da partida. Em geral, partidas aos fins de semana e feriados nacionais impõem carga menor aos servidores, possivelmente porque usuários assistem às partidas em televisores convencionais. A curva relativa à distribuição acumulada de probabilidades de um jogo realizado durante um final de semana é claramente deslocada para a esquerda. Enquanto a mediana de partidas durante a semana se encontra próximo de 90 sessões novas por segundo, a mediana da taxa de chegada de sessões de partidas durante o final de semana se encontra próximo de 40 por segundo.

Durante os jogos da copa do mundo, encontramos três níveis principais de taxa de transmissão (qualidade) de mídia pelos servidores. De acordo com a Figura 6, que apresenta a distribuição acumulada da taxa de transmissão de mídia em cada servidor separada por ISP[2], aproximadamente 40% dos usuários assistem com uma taxa de transmissão entre 150 e 400 KBps (qualidade baixa), aproximadamente 50% assistem com uma taxa de transmissão de 500 KBps (qualidade média), e aproximadamente 10% assistem com taxas de transmissão superiores a 800 KBps (qualidade alta). Usuários da *Brasil-Telecom* são

[2]Detalhes sobre localização de IPs em [de Almeida Junior et al. 2015a, de Almeida Junior et al. 2015b]

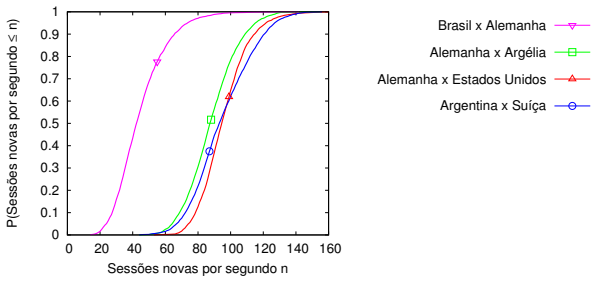


Figura 5. Taxa de criação de novas sessões em jogos representativos.

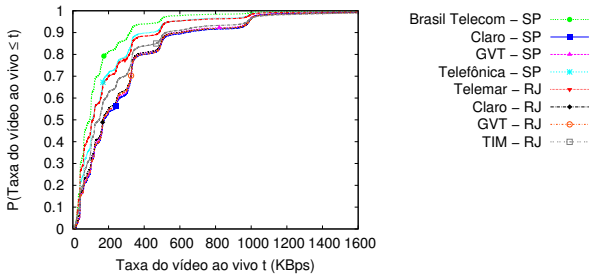


Figura 6. Taxa de transmissão da mídia ao vivo em jogos representativos.

os que assistem à mídia com a menor qualidade, enquanto usuários da Tim-RJ assistem à mídia com maior qualidade.

Finalmente, como descrevemos na seção 2.1, usuários periodicamente requisitam uma lista de reprodução. Diferente da transmissão da Copa das Confederaçõeso [de Almeida Junior et al. 2015a], as configurações para transmissão da Copa do Mundo fizeram com que os tempos entre as requisições de listas de reprodução fossem reduzidos. Verificamos que o intervalo entre as requisições de arquivos de lista de reprodução, como mostrado na Figura 7, foi bem menor do que o observado previamente, não passando de 8 segundos em 99% dos casos (o anterior chegava a 30 segundos).

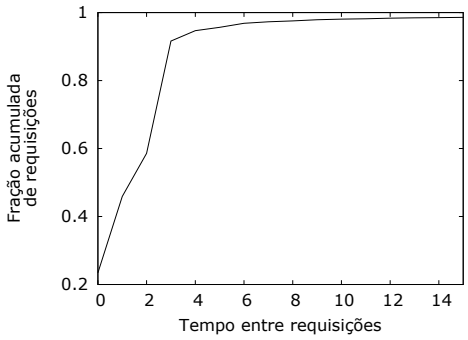


Figura 7. Distribuição do tempo entre requisições de listas de reprodução.

3. Comportamento dos clientes

Nesta seção apresentamos um modelo do comportamento dos clientes durante transmissão de vídeo ao vivo na Internet. Para cada componente do modelo, identificamos

a distribuição estatística que melhor se ajusta aos dados, dentre distribuições estatísticas amplamente usadas na literatura: normal, log-normal, exponencial, Gamma, Logística, Beta, Uniforme, Weibull e Pareto para variáveis contínuas; Poisson, Binomial, Binomial Negativa, Geométrica e Hipergeométrica para variáveis discretas. Para cada componente do modelo, os parâmetros da distribuição que mais se aproxima dos dados são determinados usando o método de estimativa por máxima verossimilhança. Após definição dos parâmetros de cada componente modelo, a distribuição com menor distância de Kolmogorov-Smirnov (distribuições contínuas) ou menor erro quadrático mínimo (LSE) (distribuições discretas) em relação aos dados é escolhida. Esta escolha também é validada com uma avaliação visual do ajuste das curvas.

3.1. Modelo do comportamento do cliente

Cada cliente estabelece uma *sessão* com os servidores de vídeo para receber e reproduzir a mídia, como mostrado na Figura 8. Durante uma sessão, o cliente requisita periodicamente a um dos servidores do provedor de conteúdo uma lista de reprodução (*pl*) indexando segmentos de mídia na codificação correspondente à qualidade que as condições de rede do cliente permitem exibir. O período de requisições de listas de reproduções e o tamanho dos segmentos de mídia são dependentes da configuração dos servidores e independem do comportamento do cliente.

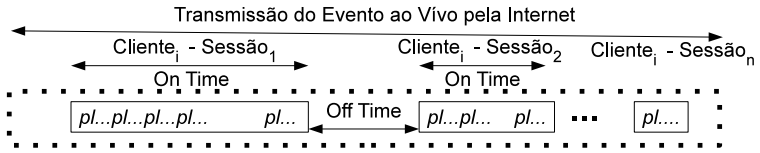


Figura 8. Comportamento de um cliente durante uma transmissão.

Cada cliente é livre para começar e interromper a visualização da mídia ao vivo a qualquer momento. A figura 8 ilustra que um cliente pode apresentar múltiplas sessões durante a transmissão de um jogo. Cada sessão tem uma duração bem definida, delimitada pelo momento em que ocorre a primeira e a última requisição a segmentos de mídia recebidas pelo provedor de conteúdo. Sessões consecutivas são espaçadas por um período compreendido entre a última requisição da sessão anterior e a primeira requisição da próxima. Por simplificação, denominamos essa duração como ON e, de forma análoga, períodos entre sessões consecutivas, denominamos por OFF.

Nós capturamos o comportamento de clientes usando um modelo "ON/OFF", como mostrado na figura 9. Neste modelo, cada cliente alterna entre o estado ativo (ON) e inativo (OFF). Durante o estado ON, o cliente está assistindo uma sessão de vídeo ao vivo. Durante o período que permanece nesse estado, cada cliente troca informações de controle com o servidor de mídia e recebe segmentos de mídia para reprodução. Após o fim de uma sessão, usuários podem voltar ao sistema de transmissão de mídia ao vivo. Nesse sentido, duas sessões ON são obrigatoriamente intercaladas por um estado de OFF. No modelo proposto, cada cliente cria uma próxima sessão, intercalada por um período OFF, com probabilidade P_{off} e pode abandonar o sistema com probabilidade $1 - P_{\text{off}}$.

O conjunto de dados que utilizamos neste trabalho não possui marcadores específicos de início e fim de sessão. Assim, nós assumimos que um usuário alterna entre um estado de atividade ON para um estado de inatividade OFF caso deixe de fazer

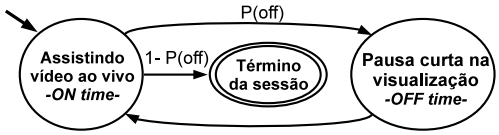


Figura 9. Modelo "ON/OFF" do comportamento de um cliente.

requisições por um período superior a um *limite de inatividade* pré-estabelecido. A escolha de um limite de inatividade apropriado é importante uma vez que valores muito pequenos podem superestimar o número de sessões, enquanto valores muito grandes podem não detectar duas sessões consecutivas que realmente existiram. Conforme apresentamos na seção 2.3, os tempos entre requisições de lista de reprodução pode ser um bom indicador desse valor limite. Nós avaliamos limites de inatividade variando entre 10 e 120 segundos. Neste intervalo, os número de sessões contabilizadas em uma partida não apresentou diferenças significativas, com diferenças médias abaixo de 10%. Dessa forma, mantivemos a metodologia de nossos trabalhos anteriores [de Almeida Junior et al. 2015a] e utilizamos um limite de inatividade de 30 segundos.

3.2. Resultados da caracterização

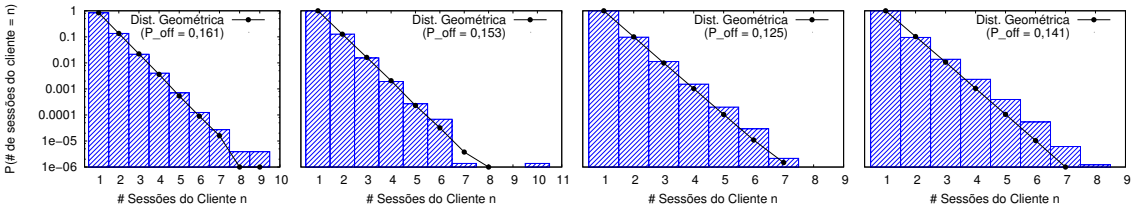
As partidas da Copa do Mundo de futebol apresentam várias nuances ao longo de seu acontecimento. Por exemplo, há um crescente número de usuários no período pré-jogo; uma saída de usuários no intervalo entre tempos e, uma saída em massa ao fim do jogo. A avaliação indiscriminada desses períodos pode adicionar ruído aos resultados. Pelo motivo exposto, nós avaliamos as características dos clientes durante um período relativamente estável. Mais precisamente, consideramos que tanto o primeiro quanto o segundo tempo da partida são relativamente estáveis, especialmente se comparados ao pré-jogo. Entretanto, escolhemos avaliar todas as amostras de sessões que se iniciaram durante o 1o. tempo da partida. Esperamos que, com essa escolha, não enviesemos os dados. Por exemplo, sessões de usuário que iniciaram próximos ao fim da partida podem gerar um número maior de sessões curtas, comparado a realidade do jogo como um todo.

A Figura 10 apresenta a distribuição do número de sessões que clientes têm ao longo de uma partida. Conforme verificamos na Figura, o comportamento entre as partidas avaliadas é muito próximo. Mais ainda, de acordo com nossas avaliações, os clientes apresentam um número pequeno de sessões. De fato, mais de 85% dos clientes tem apenas uma sessão. Apesar de existirem clientes com um número alto de sessões (e.g., 10 sessões), menos de 5% dos clientes possuem mais de duas sessões durante uma partida.

Lembre-se que, considerando o modelo ON/OFF da Figura 9, de onde nosso modelo é construído, o número de sessões por cliente pode ser computado baseando na probabilidade de existir uma transição entre o estado de atividade ON e inatividade OFF. Em outras palavras, uma distribuição Geométrica pode ser utilizada para descrever esse comportamento[3] com parâmetro $1 - P_{\text{off}}$. Na Figura 10 nós também apresentamos a distribuição geométrica de melhor ajuste aos dados apresentados. Note que as curvas e os dados medidos estão em concordância, especialmente no início da curva.

Em transmissões de vídeo ao vivo, usuários tendem a ter sessões grandes. De

[3]A função de massa de uma distribuição Geométrica com parâmetro p é $Prob(x = k) = (1 - p)^{k-1}p$.



(a) Alemanha x EUA (b) Alemanha x Argélia (c) Argentina x Suíça (d) Alemanha x Brasil

Figura 10. Número de sessões por cliente durante uma transmissão ao vivo. Uma distribuição Geométrica ajusta o nro. de sessões de um usuário. Os parâmetros dessa distribuição estão ilustrados em cada uma das figuras.

acordo com a Figura 11, praticamente 15% dos usuários tiveram sessões que cobriam toda a partida. Mais detalhadamente, as Figuras 11-a e 11-b apresentam a distribuição de probabilidade acumulada das distribuições que melhor se ajustam aos tempos ON dos 4 jogos em questão. A região em cinza, apresenta o erro padrão da média (das amostras)[4] em cada percentil da distribuição. Essa região nos dá a noção de quão distante essas distribuições estão dos dados medidos. Note que, tanto a distribuição Weibull[5], quando Gamma[6] apresentam um ajuste adequado aos dados medidos. Em ambos os casos, a diferença entre o que foi medido e o ajuste é sempre inferior a 5%.

Note também que há um número expressivo de clientes que permanecem ativos por períodos superiores ao total da partida. De fato, cerca de 10% dos tempos de ON são superiores a 100 minutos. Isso pode acontecer, principalmente, por três motivos: primeiro, os mesmos provedores responsáveis pela transmissão do conteúdo ao vivo, servem o conteúdo tradicional do provedor de serviços. Não há distinção clara para nós, nos dados coletados, a qual conteúdo cada requisição feita por clientes pertence. Assim, provavelmente há usuários assistindo a filmes ou mesmo robôs de coleta de dados em execução durante uma transmissão ao vivo da Copa do Mundo. Segundo, há transmissões relacionadas à partida mesmo após seu encerramento. Por último, entre as partidas avaliadas, há duas com prorrogação. Em especial, a partida Argentina x Suíça atraiu a atenção dos clientes de forma diferenciada.

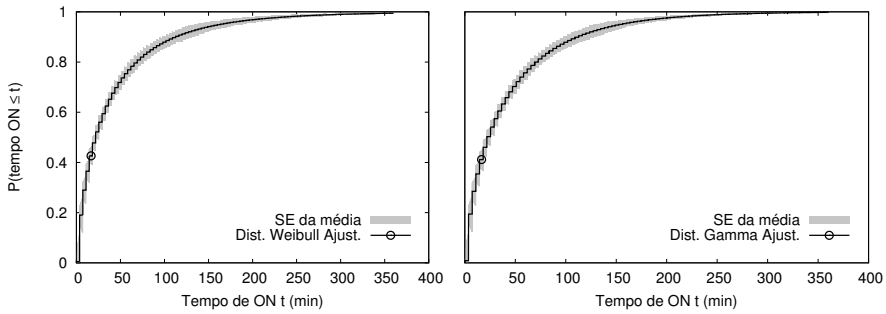
A tabela 1 detalha os parâmetros das distribuições Weibull e Gamma para cada um dos jogos avaliados, assim como o comportamento geral. Note que, a partida Argentina x Suíça destoa das demais partida. Nesse caso, enquanto a média do tempo de ON para as demais partidas ficam por volta de 34 a 38 minutos, a partida supracitada tem média de tempo de ON de 54 minutos. Além das prorrogações dessa partida, houve um crescente interesse nela, em especial, após o término do 2º tempo (Figura 4).

A Figura 12 apresenta os tempos de OFF, quando ocorrem períodos de inatividade

[4]O erro padrão da média (Standard Error of the (sample) Mean) – SEM) de um conjunto de n observações é computado como a razão entre o desvio padrão da amostra s e a raiz quadrada do número de amostras n, i.e., $s/\sqrt{n}$. Note que o $(1-\alpha)\%$ intervalo de confiança para as amostras pode ser determinado pela multiplicação do valor correspondente de SEM pelo valor do $(1-\alpha/2)$-percentil das variáveis z e t, dependendo do número de observações n [Jain 1991].

[5]Função de densidade de probabilidade (PDF) da distribuição Weibull: $p_X(x) = \frac{\alpha}{\beta}(\frac{x}{\beta})^{\alpha-1}e^{-(x/\beta)^{\alpha}}$.

[6]Gamma: $\Gamma(a) = \int_0^{\infty} s^{a-1}e^{-s}ds$.



(a) Weibull (b) Gamma

Figura 11. Tempo de duração das sessões (ON) nos quatro jogos de interesse.

Tabela 1. Distribuição do tempo de duração (ON) das sessões.

Jogo	Melhor Fitting	Média (min)	Desv. Padrão	Primeiro Parâmetro	Segundo Parâmetro
Comportamento geral	Weibull	41,93	51,94	0,700	33,708
	Gamma	41,93	51,94	0,581	0,013
Alemanha x Estados Unidos	Weibull	37,58	43,80	0,725	31,230
	Gamma	37,58	43,80	0,610	0,016
Alemanha x Argélia	Weibull	34,59	43,63	0,728	28,607
	Gamma	34,59	43,63	0,615	0,017
Argentina x Suíça	Weibull	54,33	62,66	0,743	45,987
	Gamma	54,33	62,66	0,629	0,011
Brasil x Alemanha	Weibull	38,43	51,65	0,598	26,416
	Gamma	38,43	51,65	0,472	0,012

por parte de um cliente. Assim como na Figura 11, nós apresentamos a distribuição com melhor ajuste para os dados medidos e o erro envolvido. Pela definição de atividade, que utilizamos nesse trabalho, não há tempos de OFF menores que 3 minutos. Uma distribuição LogNormal[7] é capaz de ajustar os dados medidos, com baixo erro. Note que, a grande maioria dos tempos de OFF são inferiores a 15 minutos. Mais precisamente, cerca de 85% dos tempos de OFF ocorrem por períodos que são inferiores ao tempo de intervalo que acontece no meio da partida. Intuitivamente, esse comportamento corrobora com o comportamento da carga nos servidores, observado nas Figuras 4-a e 4-b. De acordo com essas figuras, no início do intervalo nós observamos uma saída de clientes do sistema e, antes do início do 2º tempo da partida, há novamente um crescimento do número de clientes no sistema.

Finalmente, a tabela 2 resume as distribuições de cada um dos jogos, assim como a distribuição do comportamento geral. Em todas as partidas, a média dos tempos de OFF são similares. Há uma leve diferença na partida entre Alemanha x Brasil. Nesse caso, lembramos que essa partida ocorreu em um feriado nacional.

3.3. Gerador de carga sintética

Nós desenvolvemos um gerador de cargas sintéticas baseado no modelo proposto[8]. Este gerador tem por objetivo produzir um registro de acessos de usuários a um sistema de transmissão de mídia ao vivo, imitando o comportamento dos clientes reais durante um

[7]PDF da distribuição Log-Normal: $p_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{-(ln(x)-\mu)^2}{2\sigma^2}}$.

[8]Disponível em http://netlab.ice.ufjf.br.

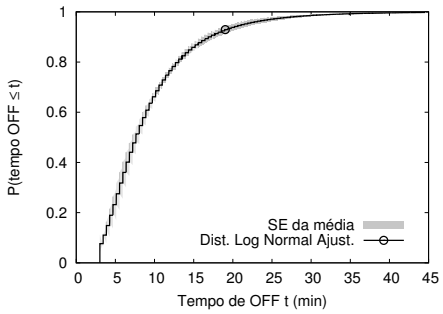


Figura 12. Tempo entre sessões (OFF) nos quatro jogos de interesse.

Tabela 2. Distribuição do tempo entre sessões (OFF).

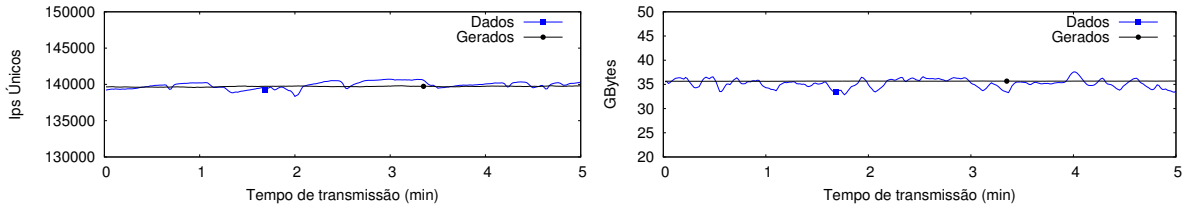
Jogo	Melhor Fitting	Média (min)	Desv. Padrão	Primeiro Parâmetro	Segundo Parâmetro
Comportamento geral	LogNormal	9,28	6,86	2,010	0,634
Alemanha x Estados Unidos	LogNormal	9,21	6,94	1,996	0,643
Alemanha x Argélia	LogNormal	9,63	6,92	2,053	0,632
Argentina x Suíça	LogNormal	9,33	6,90	2,014	0,637
Brasil x Alemanha	LogNormal	8,69	6,47	1,956	0,608

grande evento. Note que, não pretendemos nesse momento, gerar comportamentos que são dependentes do servidor. Assim, a grande variabilidade de carga notada durante uma partida não é objetivo desse simulador. Em particular, a partir de um desejado número de clientes, o gerador cria o comportamento dinâmico de cada usuário do sistema a partir das distribuições do número de sessões por partida, duração de sessão (tempo ON) e tempo entre sessões (tempo OFF). Como trabalho futuro, consideramos estender o gerador para capturar o tráfego sob a perspectiva de cada usuário. Neste caso, a escolha da qualidade da mídia e outros parâmetros dependentes do provedor de conteúdo deve ser modelada de forma mais detalhada.

Nós validamos o gerador de forma direta e indireta. Por restrições de espaço, exibiremos apenas a segunda. Ela é mais interessante pois foca em métricas que não são explicitamente modeladas pelo gerador. Para tal validação, consideramos o número de usuários de um sistema e a taxa da mídia sendo transmitida.

O gerador produziu um registro sintético com aproximadamente n usuários conectados ao sistema. Para cada um, o gerador criou o número de sessões correspondente, seus tempos de ON e de OFF. Para cada instante de tempo, nós contabilizamos o número de clientes ativos. Nós associamos uma taxa média de mídia para cada um dos n clientes ativos e contabilizamos a banda de rede para aquele momento de tempo. Nós escolhemos um período estável de um jogo real para comparar com o resultado do simulador. Escolhemos o período do minuto 25-30, do jogo Argentina x Argélia, com cerca de 140 mil clientes (figura 4). Note que, o simulador não é limitado nem direcionado a esse período.

A figura 13 apresenta a população de usuários criadas pelo gerador e a taxa de transmissão que essa população consome do provedor de conteúdo. Idealmente, o gráfico deveria apresentar os pontos próximos ao dados medidos, incluindo sua alta variabilidade. Entretanto, isto é muito difícil de se alcançar, principalmente quando o processo sendo modelado é complexo. Em particular, nosso gerador não considera variação do número



(a) Ips únicos (b) Volume (GB)

Figura 13. Validação do Gerador de Cargas Sintéticas.

de clientes nem da taxa de transmissão da mídia (figura 6), apenas usamos a média. Em suma, apesar das imprecisões, consideramos que o gerador captura o nível de carga e padrões de acesso a servidores do provedor de conteúdo de mídia ao vivo, incluindo seus tempos de atividade e inatividade.

4. Discussões e Implicações

Claramente, provedores atuais ainda não estão satisfatoriamente preparados para uma nova era onde eventos de larga escala serão regra e não mais exceção. Por exemplo, a Copa do Mundo de futebol em 2014, notoriamente um evento de grande escala, movimentou todo o Brasil. Em especial, empresas de TI se prepararam para atender a demanda por comunicação na Internet. Mesmo com todo o preparo, grandes empresas de conteúdo foram surpreendidas por demandas muito elevadas. Por exemplo, a ESPN teve picos de 1,7 milhões clientes simultâneos durante o jogo mais transmitido pela Internet até a atualidade. Toda sua infraestrutura de redes e de comunicação sofreu com essa demanda e a companhia ficou sem conexão durante grande parte da partida.[9]

Grandes eventos na Internet costumam ser bem planejados, com meses de preparação antecedente. Parte da estrutura é testada por cargas sintéticas, que tentam imitar o comportamento esperado de um cliente para aquele serviço. Em especial, para transmissão de vídeo na Internet e conteúdo web, há uma série de trabalhos que se dedicam a geração de carga sintética [Jin and Bestavros 2001, Busari and Williamson 2002, Tang et al. 2003, Krishnamurthy et al. 2006]. Apesar da importância histórica desses trabalhos, eles foram realizados em um período ainda de surgimento de aplicações com alta demanda de carga na Internet. Naquela época ainda havia poucas caracterizações acerca do comportamento de usuários e, diversas características comuns nos dias de hoje sequer existiam.

As principais caracterizações sobre o comportamento de usuários em transmissões ao vivo na Internet datam de cerca de uma década atrás. Por exemplo, [Veloso et al. 2006] avaliaram o comportamento de clientes em uma transmissão contínua de um dos primeiros *reality show* transmitido no Brasil. Nesse trabalho os autores propõem um modelo para o comportamento dos clientes e caracterizam seus componentes. Transmissões ao vivo em arquiteturas P2P se tornaram populares a partir da segunda metade da década de 2000. Nessa mesma época, surgiram caracterizações de eventos transmitidos nesse tipo de aplicação. Por exemplo, [Hei et al. 2007] estudam um sistema de transmissão ao vivo em P2P e avaliam características de seus clientes durante uma transmissão de um grande evento na China. Mais recentemente, [Borges et al. 2012] propõem um modelo

[9]http://www.businessinsider.com/espn-down-during-world-cup-2014-6

hierárquico para descrever o comportamento dos clientes no SopCast, uma aplicação popular de transmissão ao vivo em P2P. Os autores diferem o comportamento de clientes de transmissões de eventos recorrentes (e.g., novelas, jornais) e eventos únicos (e.g., finais de campeonato de jogos de futebol).

Nosso trabalho, diferente dos existentes na literatura, é focado em uma era onde transmissão ao vivo está consolidada. Nós focamos em eventos de larga escala, uma vez que tais eventos impactam os provedores de conteúdo de forma marcante. Além disso, nosso trabalho evidencia as mudanças observadas no comportamento dos clientes ao longo do tempo, deixando claro que modelos e caracterizações existentes podem estar obsoletas e, por consequência, não são apropriadas para serem utilizadas atualmente. Listamos as seguintes diferenças marcantes no comportamento dos clientes, confrontados com os principais trabalhos existentes:

- Número de sessões: o número médio de sessões de um determinado usuário está ainda menor. De fato, em transmissões de eventos em rede P2P, 40% dos clientes tinham uma única sessão [Borges et al. 2012]. Agora, cerca de 80% dos clientes têm apenas uma única sessão.
- Tempo de ON: o tempo de ON está mais longo, com médias de aproximadamente 40 min. De fato, observamos na prática a substituição de sistemas de TV tradicional por aplicativos de Internet para visualização de programação ao vivo de emissoras. Isso tem aumentado a duração de tempo de sessões dos usuários de vídeo ao vivo na Internet. Comparado a trabalhos anteriores, esse aumento é significativo. Por exemplo, durante uma partida de final de campeonato de futebol, cerca de 60% dos clientes tiveram um tempo muito curto de ON (menos de 3 min) e apenas 5% assistiram quase toda a partida [Borges et al. 2012]. Nessa mesma linha, [Hei et al. 2007] apontam que 90% das sessões duram menos que 10 min.
- Tempo de OFF: os tempos de inatividade também estão alterados atualmente. Enquanto em [Borges et al. 2012], 35% dos clientes apresentavam tempos de OFF superiores a 20 minutos, no trabalho atual, apontamos que apenas cerca de 6% dos usuários ficam inativos pelo mesmo período.

5. Conclusões e Trabalhos Futuros

Neste artigo, apresentamos uma caracterização e modelagem dos clientes de um sistema de vídeo ao vivo durante um evento de larga escala na Internet. O estudo foi feito a partir de cargas de trabalho da transmissão das 64 partidas de futebol da Copa do Mundo de 2014, fornecidas por um dos principais provedores de conteúdo do Brasil. Para analisar o comportamento dos clientes foi proposto um modelo simples que captura as principais características de um cliente, como sua forma de interação com os provedores de conteúdo, seu tempo de sessão e seus períodos de inatividade.

Quanto ao comportamento dos clientes, observamos que há pouca variação entre os jogos avaliados, o que sugere que as distribuições encontradas descrevem bem seus comportamentos. A maioria das sessões são longas, e a grande maioria dos clientes apresenta uma única sessão. Além da diferença das distribuições que descrevem alguns comportamentos dos clientes, encontramos também diferenças qualitativas significativas marcantes. Por exemplo, o tempo de sessão é notoriamente maior durante os eventos atuais, assim como o tempo de inatividade, quando existente, se reduziu. Isso indica que os

padrões de carga de eventos são significativamente distintos dos existentes na literatura.

Trabalhos futuros incluem extensões para novos conjuntos de dados e a extensão do gerador de cargas sintéticas para outros aspectos da carga de trabalho, por exemplo, taxa de chegada de clientes e adaptação da taxa de transmissão da mídia. Mais ainda, dada a natureza dos eventos, com grandes volumes de dados, pretendemos criar mecanismos para descrever o comportamento da carga de trabalho a partir de um número pequeno de variáveis.

Referências

Barroso, L. A., Clidaras, J., and Hölzle, U. (2013). The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154.

Borges, A., Gomes, P., Nacif, J., Mantini, R., de Almeida, J. M., and Campos, S. (2012). Characterizing SopCast Client Behavior. *Comput. Commun.*, 35(8):1004–1016.

Busari, M. and Williamson, C. (2002). ProWGen: A Synthetic Workload Generation Tool for Simulation Evaluation of Web Proxy Caches. *Comput. Netw.*, 38(6):779–794.

Cesario, M. V. (2012). Uso de anycast para balanceamento de carga na globo.com. *Talks and Tutorial, SBRC 2012*. Disponível em: `http://pt.slideshare.net/marcuscesario/apresentacao-anycast-sbrc201205`.

de Almeida Junior, W., Almeida, B., Ítalo Cunha, Almeida, J. M., and Vieira, A. B. (2015a). Caracterização do Tráfego e Impacto de Rede da Transmissão de um Grande Evento Esportivo. In *33o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC*.

de Almeida Junior, W., Borges, A. V., and Ítalo Cunha (2015b). Avaliação de transmissão ao vivo de grandes eventos pela internet. *PPGCC-UFJF - Dissertação de mestrado*. Disponível em: `http://www.ufjf.br/pgcc/files/2014/06/WagnerAlmeida.pdf`.

Gebert, S., Pries, R., Schlosser, D., and Heck, K. (2012). *Internet access traffic measurement and analysis*. Springer.

Hei, X., Liang, C., Liang, J., Liu, Y., and Ross, K. W. (2007). A measurement study of a large-scale p2p iptv system. *Multimedia, IEEE Transactions on*, 9(8):1672–1687.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons.

Jin, S. and Bestavros, A. (2001). GISMO: A Generator of Internet Streaming Media Objects and Workloads. *SIGMETRICS Perform. Eval. Rev.*, 29(3):2–10.

Katabi, D. and Wroclawski, J. (2000). A framework for scalable global ip-anycast (gia). *ACM SIGCOMM Computer Communication Review*, 30(4):3–15.

Krishnamurthy, D., Rolia, J. A., and Majumdar, S. (2006). A Synthetic Workload Generation Technique for Stress Testing Session-Based Systems. *IEEE Trans. Softw. Eng.*, 32(11):868–882.

Tang, W., Fu, Y., Cherkasova, L., and Vahdat, A. (2003). MediSyn: A Synthetic Streaming Media Service Workload Generator. In *Proc. of the NOSSDAV*, pages 12–21.

Veloso, E., Almeida, V., Jr., W. M., Bestavros, A., and Jin, S. (2006). A Hierarchical Characterization of a Live Streaming Media Workload. *IEEE/ACM Trans. Netw.*, 14(1):133–146.

Trilha Principal do SBRC 2016
Sessão Técnica 12
Sistemas Distribuídos

Supporting Dynamic Reconfiguration in Distributed Data Stream Systems

Rafael Oliveira Vasconcelos[1,2], Igor Vasconcelos[1,2], Markus Endler[1], Sérgio Colcher[1]

[1]Department of Informatics, Pontifical Catholic University of Rio de Janeiro (PUC-Rio) Rio de Janeiro, Brazil

[2]University Tiradentes (UNIT), Aracaju, Brazil

{rvasconcelos,ivasconcelos,endler,colcher}@inf.puc-rio.br

Abstract. *The design and development of adaptive systems brings new challenges since the dynamism of such systems is a multifaceted concern that ranges from mechanisms to enable the adaptation on the software level to the (self-) management of the entire system using adaptation policies or system administrators, for instance. Data stream processing is one example of system that requires dynamic reconfiguration. While dynamic reconfiguration is a desirable feature, stream systems may suffer with the disruption and overhead caused by the reconfiguration. In this paper, we propose and validate a non-disruptive reconfiguration approach for distributed data stream systems that support stateful components and intermittent connections. We present experimental evidence that our mechanism supports safe distributed reconfiguration and has negligible impact on availability and performance.*

1. Introduction

An adaptive system must be able to cope with changes in its execution environment and in the requirements that it must comply with [Ma et al. 2011]. The authors [Ma et al. 2011] further emphasize that changes are hard to predict at design time. On the other hand, many distributed systems have to provide services for 24x7, with no downtime allowed [Giuffrida et al. 2014]. This continuous execution makes it difficult to fix bugs and add new required functionality on-the-fly as this requires non-disruptive replacement of parts of a software version by a new ones [Ertel and Felber 2014]. The authors in [Ertel and Felber 2014] further explain that prior approaches to dynamic reconfiguration (a.k.a. dynamic adaptation, live update or dynamic evolution) require the starting of a new process and the transfer of states. However, [Hayden et al. 2012] argue that the cost of redundant hardware, and the overhead to transfer and synchronize the system's state may be considerable high.

Despite extensive research in compositional dynamic software adaptation [Vasconcelos et al. 2014] [Kakousis et al. 2010] (i.e., the sort of reconfiguration addressed by our work), safe reconfiguration is still an open problem [Giuffrida et al. 2014]. A common approach is to put the component that has to be updated into a safe state, such as the quiescent state [Kramer and Magee 1990], before reconfiguring the system [Ghafari et al. 2012]. Thus, a safe reconfiguration must drive the system to a consistent state and preserve the correct completion of on-going activities [Ma et al.

2011]. At the same time, dynamic reconfiguration should also minimize the interruption of system's service (i.e. disruption) and the delay with which the system is updated (its timeliness). In [Ertel and Felber 2014], the authors also explain that orchestrating the restart of all the components is very challenging if the system's service must not be interrupted.

Aligned with the aforementioned requirements, applications in the field of data stream processing require continuous and timely processing of high-volume of data, originated from a myriad of distributed (and possibly mobile) sources, to obtain online notifications from complex queries over the steady flow of data items [Stonebraker et al. 2005] [Cugola and Margara 2012]. Thus, while dynamic reconfiguration is a desirable feature, stream systems may suffer with the disruption and overhead caused by the reconfiguration. Furthermore, at the same time, it is not feasible to block (or await a quiescent state) some of the involved components.

In order to address the aforementioned issues, we propose and validate a non-disruptive approach for dynamic reconfiguration that preserves global system consistency in distributed data stream systems. Such approach should (i) handle nodes that may appear and disappear at any time, (ii) ensure that all data items (of the data stream) are processed exactly once, and (iii) do not disrupt the system to perform a reconfiguration. More specifically, the main contributions of this work include a mechanism to enable safe reconfiguration of distributed data stream systems, a prototype middleware that implements the mechanism, and experiments that validate and demonstrate the safety of the proposed solution.

The remainder of the paper is organized as follows. Section 2 presents an overview of the key concepts and system model used throughout this work, as well as introduces a motivating scenario. Section 3 delves into details the proposed approach to dynamic reconfiguration in multiple distributed instances. Section 4 summarizes the main results of the assessment conducted. Finally, Section 5 reviews and discusses the central ideas presented in this paper, and proposes lines of future work on the subject.

2. Fundamentals

This section presents our system model and related works, as well as a motivating scenario and the main concepts about data stream processing.

2.1. Related Work

A common problem in the field of software reconfiguration at runtime is the identification of states in which the system is stable and ready to evolve [Ma et al. 2011]. The authors [Ertel and Felber 2014] propose a framework for systems that are modeled using (data)flow-based programming (FBP) [Morrison 2010]. The idea behind FBP is to model the system as a directed acyclic dataflow graph where the operators (vertices) are functions that process the data flow and the edges define de input and output ports of each operator. Since the messages are delivered in order, this proposal forwards special messages informing when a component (a.k.a. operator) is safe to be reconfigured. When a component receives such message, it is substituted by the new version. Despite the advantages of such proposal, it neither handles nodes that may appear or disappear any time nor assumes that source components (i.e., components that

generate data) may have different versions. Therefore, the problem with the work [Ertel and Felber 2014] is that either all components will perform the reconfiguration or none of them can proceed with the reconfiguration, similar to a transaction. As demonstrated by [Nouali-Taboudjemat et al. 2010], such approach does not have a good convergence rate for systems that deal with mobile devices. Finally, while a component is updated to its new version, it is unable to process the data flow, thereby causing a system disruption (or, at least, a temporary contention).

In the subject of dynamically reconfigurable stream processing systems, the work [Schweppe et al. 2010] proposes a method for flexible on-board processing of sensor data of vehicles that is modeled as a data stream processing system. In regards to dynamic reconfiguration issues, this approach is able to change component's parameters, system's topology (i.e., the graph structure), or how the component stores its data. However, the work does not deal with consistency for topology changes. Furthermore, the proposal does not allow the modification of a component by its new version (i.e., it does not permit compositional adaptation). Finally, the system targeted by the authors is deployed on a single node (i.e., a car), conversely from ours that is distributed and has multiple instances of a component type.

While some works [Vandewoude et al. 2007] block the system to enable its evolution, others [Ma et al. 2011] [Ghafari et al. 2012] [Chen et al. 2011] are capable of executing the old and new versions concurrently. The proposals in [Ma et al. 2011] [Ghafari et al. 2012] ensure that, while a reconfiguration is performed, any extant transaction with all its sub-transactions is entirely executed in the old or in the new system's configuration (i.e. old or new versions) [Ma et al. 2011]. The major drawback is the overhead required to maintain the graph representing the system's configuration [Ghafari et al. 2012]. With the aim of solving the latter drawback, the authors in [Ghafari et al. 2012] chose to use *evolution time* (i.e., the timestamp in which a reconfiguration is performed) as a mechanism to decide if a transaction should be served by the old or new version. Thus, a transaction initiated before the evolution time is served by the old version, otherwise it is served by the new version. Although the evolution time causes minor impact in the system's performance, time synchronization in distributed systems is a well-known problem for such systems.

To the best of our knowledge, no other research work copes with the problem of adapting at run-time a component type that has multiple distributed instances spread over the distributed data stream processing system while guaranteeing the system consistency. That is, each component type may be deployed over many distributed (and possibly mobile) nodes rather than on a single node. Thus, the modification to a new version of one component type requires the coordinated modification of many component instances deployed over numerous (mobile) nodes.

2.2. Motivating Scenario

Consider a data streaming application that collects some sensor data from a smartphone, for instance, and sends it to a distributed node (e.g., cloud) in order to process the data (Figure 1). In typical stream applications in the field of mobile social applications [Ganti et al. 2011], people share sensor information about their daily physical exercise among friends and acquaintances. As an example, BikeNet [Eisenman et al. 2007] probes

location and bike route quality data, such as CO_2 level and bumpiness of the road during the ride, in order to share information about the most suitable routes [Ganti et al. 2011]. We may decompose such applications using the components shown in Figure 1.

As the system may have an arbitrary number of mobile nodes and processing hosts or servers (in the cloud), each component in Figure 1 will typically have many instances that are deployed at different nodes. As illustrated in Figure 1, the components *Data Gathering* and *Pre-Processing* run on the mobile nodes (e.g., smartphones), the *Processing* and *Post-Processing* components run in the servers of a cloud, while the components *Sender* and *Receiver* run on both mobile nodes and servers. Therefore, in order to replace a component, the reconfiguration platform has to guarantee distributed consistency. Figure 1 illustrates the component types deployed on each node and that each component type has several distributed instances spread in the distributed system. We consider that each step in the processing flow has an arbitrary number of servers that share their workload and that data sent by a client can be forwarded to any server at step A, for load balancing purposes.

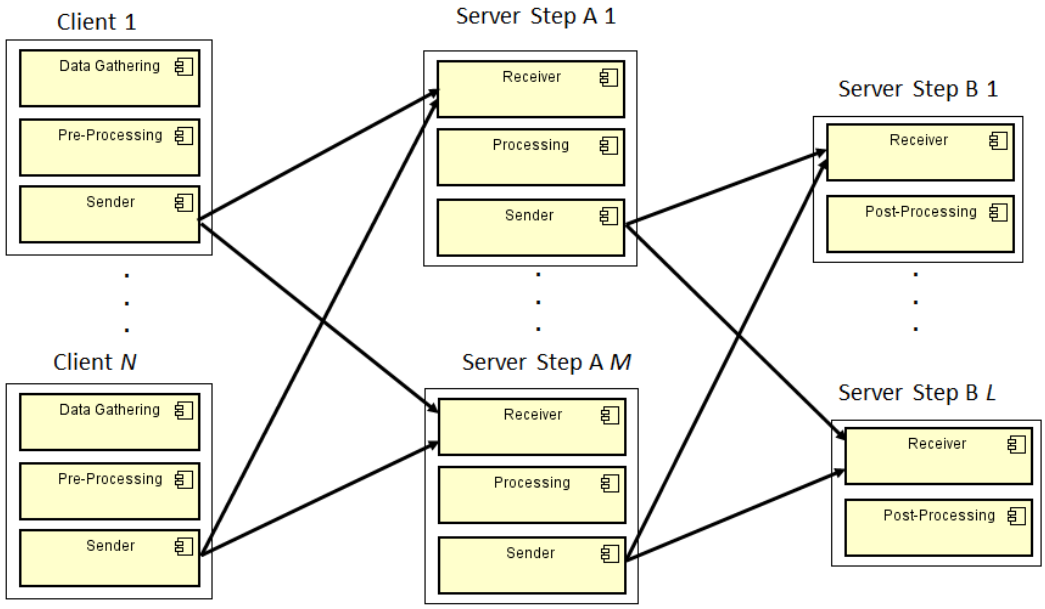


Figure 1. Deployed components and the system data flow

2.3. Data Stream Processing

Data stream processing is a computational paradigm [Schweppe et al. 2010] that is focused at sustained and timely analysis, aggregation and transformation of large volumes of data streams that are continuously updated [Cugola and Margara 2012]. A data stream is a continuous and online sequence of unbounded items where it is not possible to control the order of the data produced and processed [Babcock et al. 2002]. Thus, the data is processed on the fly as it travels from its source node downstream to the consumer nodes, passing through several distributed processing nodes [Cherniack et al. 2003], that select, classify or manipulate the data. This model is typically represented by a graph where vertices are source nodes that produce data, operators that implement algorithms for data stream analysis, or sink nodes that consume the processed data stream, and where edges define the data paths among the nodes (i.e., stream channels).

Therefore, many stream processing systems are inherently distributed and may consist of dozens to hundreds of operators distributed over a large number of processing nodes [Cherniack et al. 2003], where each processing node executes one or several operators. Furthermore, such systems have an *execution dependency* among component types and each component type may have multiple instances. The execution dependency differs from deployment dependency in that it considers the runtime data and control flows, instead of the static dependencies [Ma et al. 2011] among software modules.

The work [Stonebraker et al. 2005] proposes eight general rules that hold for data stream processing; but the most important rules related to our work are: "keeping the data moving", "generating predictable outcomes", "guarantying data safety and availability", and "processing and respond instantaneously". In order to satisfy these rules, a system reconfiguration must be reliable (i.e., it does not produce erroneous outcomes and affect the system's availability), and not disrupt or block the stream. In spite of the recognized importance of dynamic reconfiguration for such systems [Babcock et al. 2002], researchers [Schweppe et al. 2010] confirm that most data stream processing middleware have no compositional adaptation mechanisms.

2.4. System Model

Our notion of a stream processing system, inspired by [Schweppe et al. 2010], is a directed acyclic graph that consists of multiple operators (i.e., components) deployed at distributed device nodes. More formally, the graph G = (V, E) consists of vertices and edges. A vertex represents an operator and an edge represents a stream channel. An edge e = (*v1*, *v2*) interconnects the output of vertex *v1* with the input of vertex *v2*. Vertices without input ports (i.e., without incoming edges) are referred as source vertices. Correspondingly, vertices without output ports are called sink vertices. Finally, vertices with both input and output ports are called inner vertices. A tuple t = (*val*, *path**) consists of a value (*val)* and an execution path (*path*)* that holds the operators, and their versions, that a tuple t traveled through G. For instance, a tuple t that traveled from source vertex SO1 to sink vertex SI1 via operators O1 and O2 holds *path* = {SO1, O1, O2}. A stream s = (t*) between *v1* and *v2* consists of an ordered sequence of tuples t* where t1 < t2 represents that *t1* was sent before *t2* by a node *n1*. A vertex is composed of f^{select}, f^{output} and f^{update} functions. When a vertex *v1* generates a tuple (i.e., sends it via the output port), its succeeding vertices (i.e, the vertex that receive the stream from *v1*) receives such tuple via the function f^{select}, which is in charge to select, or not, this tuple to be processed by the function f^{update}.

In order to standardize the terms and notations used throughout this work, an operator (a.k.a. graph vertex) will be generically referred to as a component. A node is any physical device node (e.g., desktop and smartphone) that executes a component. A Processing node (PN), in turn, is a node that holds at least one inner operator (i.e., an operator with input and output ports). Furthermore, as data stream systems must be elastic to adapt to variations in the volume of the data streams [Vasconcelos et al. 2013], we consider that some processing nodes (PNs) share their workload, as shown in Figure 1 where data originated by a client may be delivered to any server at step A.

As the main assumptions about the system and network, we assume that any node is able to enter or leave the system anytime, messages are reliably delivered in

FIFO (First In, First Out) order between two nodes. We also assume that nodes are able of correctly executing their components, and that components *per se* do not introduce any flaws that may lead the system to an inconsistent state. We consider that there are no Byzantine failures and if any node fails (fail-stop), the system is able to timely detect the node's failure.

3. Distributed Dynamic Reconfiguration

This section presents our approach to enable dynamic reconfiguration in distributed stream processing systems. The proposed approach is based on the idea that a data produced by a Client Node (CN) has to be entirely processed by a specific version of each component. However, there is no problem in updating a component C while a tuple T traverses the system as long as C is multi versioned (i.e., C has a version C1 that represents the old version and C2 that represents the new version) and T is exclusively processed by C1 or C2. Differently from other works, our proposal does not need to wait for the system to reach a quiescent state (or safe state) to reconfigure a f^{update} function. In the scenario shown in Figure 1, there is a dependency between the Sender and Receiver components since they have to use a compatible algorithm in order to exchange messages (a.k.a. tuples) through the network. In this way, if the reconfiguration of the Receiver happens before any client sends a message, all messages are processed by the new version of the Receiver component since the clients use the new version of the Sender component. However, if the reconfiguration happens while the clients send messages (i.e., the data stream has a continuous flow), some messages must be processed by the new version (if and only if – iff – the message was sent by the new version of the Sender) whereas others have to be processed by the old version. At this time in which there are some clients with the old version and others with the new one, the servers must have deployed both versions of the component to be able to receive correctly the messages from any client. Therefore, both versions of the Receiver component must coexist at the servers while the system is being reconfigured.

Each component has one or more f^{select}, f^{update} and f^{output} functions and components have interdependencies. The advantage of enabling a component to have more than one f^{update} function executing concurrently is that, in face of a reconfiguration, the new function is able to process part of the data stream while the old one is still in use and thus cannot be deactivated. Accordingly, when a tuple T is received by an f^{select} function, it has to choose the correctly f^{update} to process T. To do so, the f^{select} function verifies the *path* of T. The f^{select} and f^{output} represent the input and output ports, respectively, of a component, whereas the f^{update} is the algorithm in charge of processing the transformation on the incoming data stream. Thus, we are able to reconfigure the algorithms that process the data streams (i.e. f^{update} functions) and the system's topology by means of reconfiguring the f^{select} and f^{output} functions.

In many systems, such as data stream processing ones, the components have indirect dependencies. An indirect dependency is a mutual dependency [Ertel and Felber 2014] between components *X* and *Z* in which they are not directed interconnected to each other (i.e., there is no edge interconnecting them). In other words, they are not directly interconnected. Thus, there is at least one inner component *Y* to enable a tuple from X arrives at *Z*. In our example (Figure 1 and Figure 2), the Processing component depends on Pre-Processing component; however, there are two components between them. Due

to the possibility of multi versioned components and indirect dependencies, each tuple holds the execution path to enable our proposal to maintain the system consistency. Yet, Receiver depends on Sender and Post-Processing depends on Processing.

3.1. Dependency Management

In our example of the data stream system, the Processing's f^{select} function has to know the version of the Pre-Processing's f^{update} applied to pre-process a tuple T to avoid inconsistency. Figure 2 shows the partial data flow of a tuple T when the system has the f^{update} functions A1, D1 and E1 of Pre-Processing, Processing and Post-Processing components, respectively. Figure 3 shows that the versions A2, D2 and E2 were added to the system and that Processing D1 (i.e., the Processing's f^{update} function D1) and Post-Processing E1 processed the tuple T in order to maintain the system consistency. Thus, when T arrives at the Processing's f^{select} function (Figure 3), it verifies that T comes from Pre-Processing A1 and then uses the Processing D1 to process T. The same happens at Post-Processing. Thus, every component has to be aware of its dependency to be able to choose the right f^{update} function.

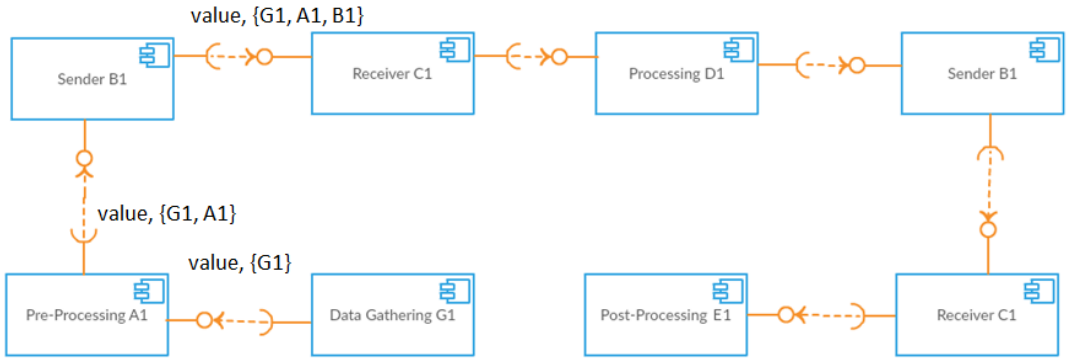


Figure 2. Partial data flow of the motivating scenario where data is to be received by Receiver C1

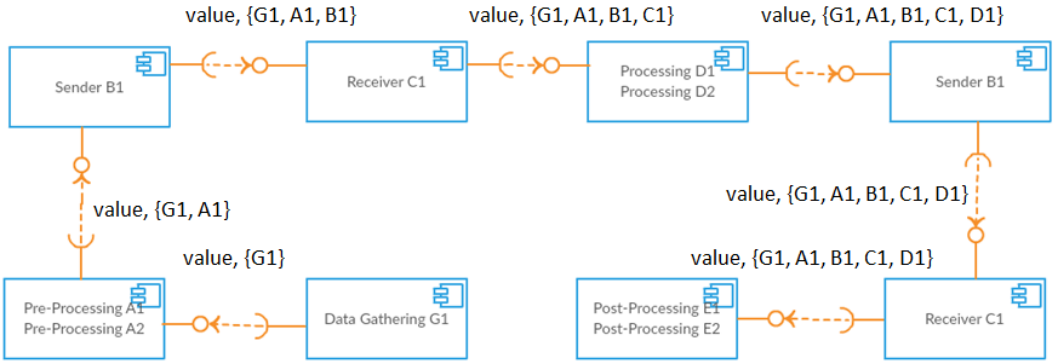


Figure 3. Tags of the data in a partial reconfigured system

The dependencies can be managed using two approaches, static or dynamic dependency management. The former, which is the simplest one, does not take into account the "downstream" dependent components to generate the execution path of a tuple. Thus, whenever a component processes a tuple, the f^{update} function's version of such component is added into the tuple's execution path, as illustrated by Figure 2 and Figure 3. Conversely, the latter approach verifies if there is any dependent component

before adding the version of the f^{update} function into the execution path. If there is no dependent component, the version is not added into the execution path. Furthermore, at each component, the execution path is evaluated to check and discard the versions that have no more dependent components.

3.2. Distributed Multi Instance Reconfiguration

So far, we have not discussed how to adapt distributed multi instances of component types. As shown in Figure 1, each component type may have many instances deployed over the distributed nodes, such as the Processing component type that is deployed on all servers at step A. If one needs to change the Pre-Processing and Processing component types, the old/new version of the Processing instance can only process data originated from the old/new version of the Pre-Processing instance, as mentioned before. However, considering that the servers share the workload and our reconfiguration proposal cannot forward the data stream to a specific server, the system has to keep the old Processing instances while there is some old Pre-Processing instance to ensure that all data stream originated from the old Pre-Processing instances are processed by the old Processing instances. Furthermore, the new Processing instances must be deployed before the new Pre-Processing instances. Thus, we have to coordinate the reconfiguration execution at all nodes.

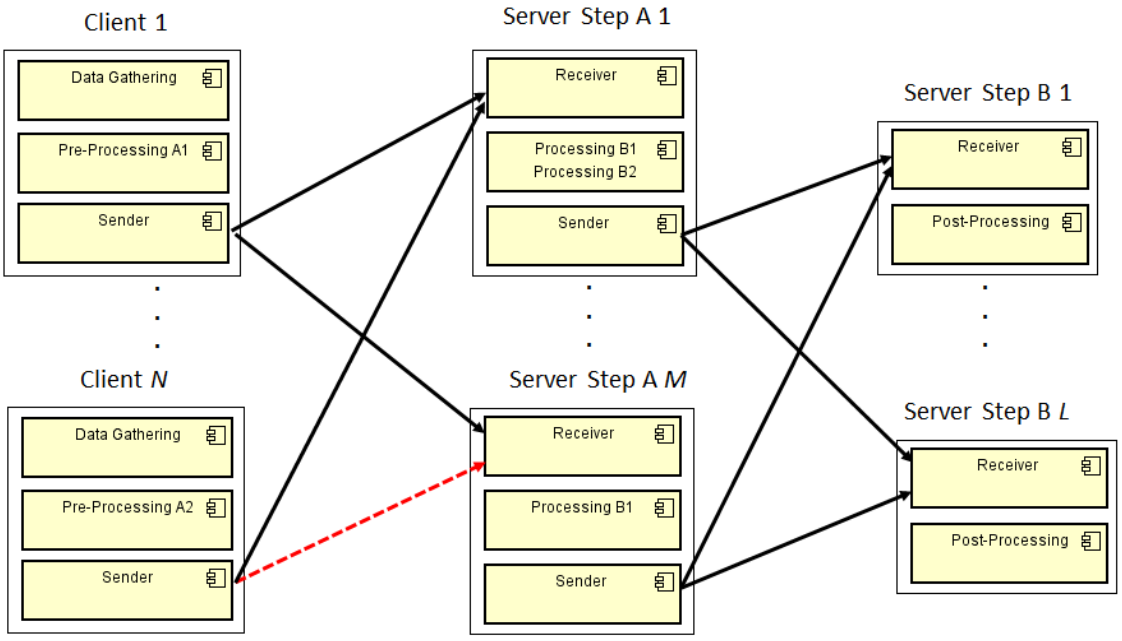


Figure 4. Partial inconsistent reconfiguration

In Figure 4, if part of the data stream from Client N goes to Server Step A M, the system achieves an inconsistent state since the server is unable to properly process the data stream. Thus, the servers must have both versions (i.e., Processing B1 and B2) while the system is partially reconfigured because some clients are not yet reconfigured. As soon as the clients are reconfigured, and there are no tuples in transit from Pre-Processing A1, the Processing B1 instances are removed from the servers at step A and the reconfiguration terminates, as shown in Figure 5. Therefore, our approach guarantees that the servers are able to handle data stream from any client, reconfigured or not.

In order to safely remove a component R and to guarantee that there are no tuples in transit towards R, we have borrowed ideas from the seminal papers [Lamport 1978] [Chandy and Lamport 1985] to have the notion of computation time in a decentralized manner. The idea is to add a special message (called *marker*) into the data stream to mark a specific time T in which a component is safe to be removed. Similarly to [Ertel and Felber 2014], as soon as R receives markers from all its dependent component instances, R is removed from the system. Thus, in order to remove Processing B1, whenever a client removes Pre-Processing A1, it adds a marker into its *M* stream channels to servers at step A, and whenever *N* markers (i.e., one marker from each client) arrives at a server at step A, Processing B1 is removed from such server. Then, Processing B1 is safely and gradually removed from the entire system.

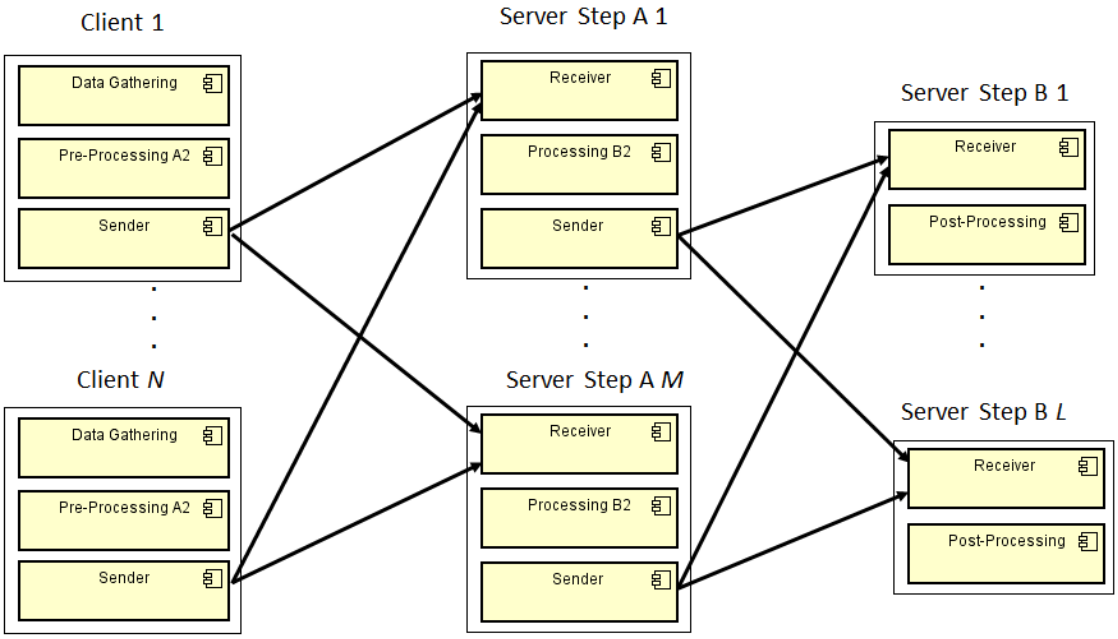


Figure 5. Reconfigured system

As in our example clients are expected to be mobile nodes using mobile networks (e.g., 3G and 4G), they may lose their connection any time. Our proposal handles nodes that may appear or disappear as follows. While the system is being reconfigured, if a client node X becomes unavailable before updating its Pre-Processing instance to version A2, the servers at step A should postpone the removal of Processing B1 until X becomes available and finishes its reconfiguration. We could adopt other strategies such as a timeout to consider that X leave permanently the system. However, the discussion about reconfiguration policies is not the focus of this work. Thus, we choose to adopt a pessimistic approach and guarantee that all tuples must be properly processed by the correct component version.

Although the system is partially reconfigured while client node X does not finish its reconfiguration, the system is able to evolve to Pre-Processing A3 and Processing B3. This is possible because each component is able to have many f^{select}, f^{update} and f^{output} functions. Thus, the Pre-Processing and Processing component types are able to have the f^{update} functions A1, A2 and A3, and B1, B2 and B3, respectively. After the second

reconfiguration from A2 and B2 to A3 and B3, the versions A2 and B2 are removed, considering that this second reconfiguration have finished.

4. Evaluation

In this section, we present the evaluation results regarding our approach. Using our prototype implementation, we have measured the time required to reconfigure the system (i.e. update time) and the disruption caused by the reconfiguration varying the number of CNs and the rate (i.e. frequency) of tuple production, as well as the overhead imposed by our approach. Finally, concerning mobile CN disconnections, we have emulated CN disconnections to verify the amount of time required to complete a reconfiguration after an MN becomes available again.

Our hardware test was composed of a Dell Laptop Intel i5-3210M 2.5GHz, 8GB DDR3 1333MHz and Wi-Fi (802.11n) interface running Windows 8.1 64 bit, a Dell Laptop Intel i5 M 480 2.66GHz, 8GB DDR3 1333MHz and Wi-Fi (802.11n) interface running Ubuntu 12.04 LTS 64 bit, and a Fast Ethernet router with 802.11n wireless connection. We used one laptop to emulate the CNs and the other one to run the PNs. Our prototype application used for evaluation has been implemented using the Java programing language and SDDL (Scalable Data Distribution Layer), a middleware for scalable real-time communication [Silva et al. 2012].

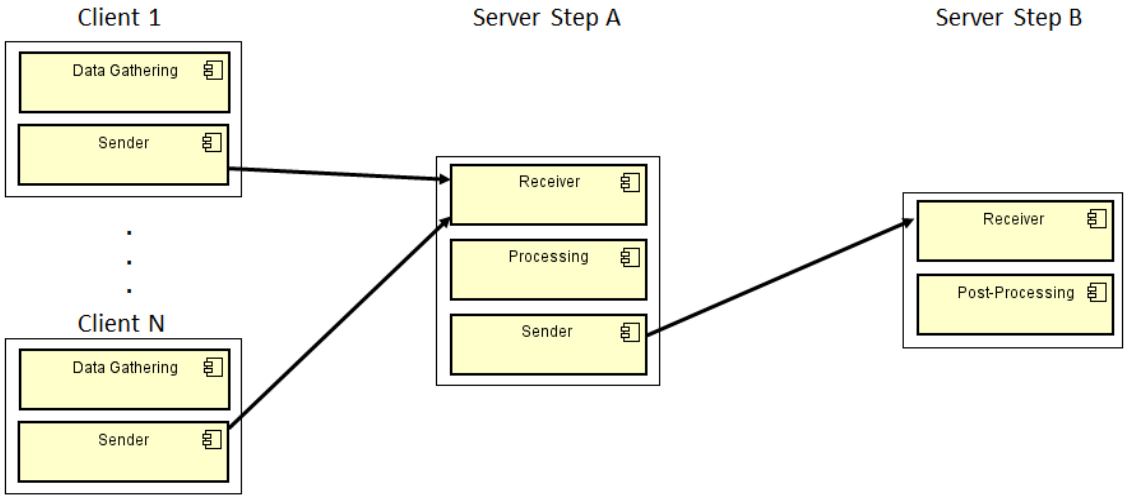


Figure 6. Deployed components and validation application data flow

The evaluation application, shown in Figure 6, is a simplification of the one presented in the motivating scenario in which there is no Pre-Processing component at the CN. The Data Gathering component generates mock data that is encoded and sent by the Sender, while the Receiver decodes and forwards the data to the Processing component. Data Gathering generates pre-defined strings that are later concatenated with other pre-defined strings at the Processing component. The Post-Processing then checks if the received string is valid or not. As we reconfigure the f^{update} functions of the Sender and Receiver to use a different encoding/decoding method, the tuple's value would be inconsistent at Post-Processing if our prototype did not properly handle such reconfiguration. Figure 6 illustrates how the components were deployed at each node

where client is a CN and server is a PN. The reconfiguration performed is changing the encoding/decoding method from ISO-8859-1 to UTF-16.

4.1. Experiments

In order to measure the update time and the service disruption, we varied the number of CNs from one to 1,000 and the system's tuple production rate from 10 tuples/s (tuples per second) to 10,000 tuples/s. We emulated each scenario 10 times, totalizing 160 emulation runs and the confidence level for all results is 95%. The JAR file that encapsulates each deployed component has 3KB (kilobytes). We also evaluated the state management overhead and the overhead that our prototype imposes while no reconfigurations are performed. All experiments were done using the static approach of the dependency management.

Regarding to reliability of the reconfiguration approach, all reconfigurations were performed consistently. This means that all tuples were *processed exactly once* and were handled by the right f^{update} function at the Receiver. Thus, we were able to achieve the global system consistency.

4.1.1 Update Time

The update time experiment measured the Round-trip Delay (RTD), which encompasses the time interval from the instant of time the reconfiguration starts until all CNs complete the execution of the reconfiguration. The tuple production rate informs the production rate of the entire system, not for each CN.

As expected since our approach does not need to await for a safe state to proceed the reconfiguration, the update time is considerably stable. It ranges from 10.05ms in the scenario with 1 CN and production rate of 10 tuples/s to 11.88ms in the scenario with 1,000 CNs and 1,000 tuples/s. The update time starts increasing with a tuple production rate of 10,000 tuples/s; however, this time increase could be affected by the high CPU (central processing unit) usage. With such production rate, both laptops experienced CPU usages of 100%. This scenario has to be revaluated to show if the update time increase is due to the production rate or due to the laptops' overload.

4.1.2 Service Disruption

In the service disruption experiment, we show the impact that a reconfiguration causes on the system's throughput (of tuples). The goal was to identify if the throughput at the end of the stream processing was affected by the reconfiguration performed. Thus, in order to measure the service disruption, we assess the system's throughput at the Post-Processing component while the tuple production rate was of 10,000 tuples/s.

According to our experiment results, the service disruption was negligible. The throughput in in the scenarios with 1 and 100 CNs had a minor increase in the reconfiguration time T when compared with $T - 1$, from 9,996 tuples/s to 10,001 tuples/s and from 10,000 tuples/s to 10,013 tuples/s, respectively. In both scenarios, the throughput was not affect by the reconfiguration. In the scenarios with 10 and 1,000 CNs, on the other hand, the reconfiguration had an insignificant influence when we compare the throughput at time T and $T - 1$. With 10 CNs, the throughput varied from 10,010 tuples/s at $T - 1$ to 10,008 tuples/s at T, which represents only 0.02% of service

disruption. In the same way, the throughput varied from 9,999 tuples/s to 9,992 tuples/s, which represents a decrease of 0.07%, in the scenario with 1,000 CNs. The experiments demonstrates that our approach causes, in some cases, a marginal decrease (lower than 0.1%) in the system's throughput.

4.1.3 Overhead

We measured the overhead that our mechanism imposes on the application prototype when no reconfiguration is performed. To do this experiment, we calculated the time required by the application to generate and process 100,000 tuples with and without the reconfiguration prototype. The overhead was 2.32%, which seems to be reasonable for the most of the applications considering the functionality provided by our proposal.

4.1.4 CN Disconnection

Due to the possibility of disconnections of mobile CNs, we assessed the amount of time required to complete a reconfiguration after an MN becomes available again. To do so, we have forced a CN to disconnect before the reconfiguration and reconnect after the reconfiguration. The reconnection time encompasses the time interval from the instant of time the CN reconnects until it completes the execution of the reconfiguration. As the number of CNs and the tuple production rate has minor impact on the update time (see Section 4.1.1), we conducted this experiment with 1,000 CNs and 1,000 tuples/s. As soon as the CN reconnects, it took 31.50ms to complete the reconfiguration.

5. Conclusion and Future Work

This paper proposes and validates a non-quiescent approach for dynamic reconfiguration that preserves system consistency in distributed data stream systems. Unlike many works that require blocking the affected parts of the system to be able the proceed a reconfiguration, our proposal enables the system to evolve in a non-disruptive way. Apart from the consistency, our proposal handles nodes that may disconnect and reconnect at any time. Hence, the main contributions of this paper are (i) a mechanism to enable safe reconfiguration of distributed data stream systems, (ii) a prototype middleware that implements the mechanism, and (iii) experiments that validate and demonstrate the safety of our proposal.

Problems such as parametric variability and reconfiguration reasoner, which is responsible for deciding when an reconfiguration is required, which alternative best satisfies the overall system goal, and which reconfigurations are needed in order to drive the system to an optimal state are not covered by our research.

We are aware that much work and research is still needed; however, considering the encouraging preliminary performance evaluation, we are confident that our approach will facilitate the development of reconfigurable stream systems. In a scenario with 1,000 CNs and 1,000 tuples/s, our prototype was able to reconfigure the entire system in 16.65ms, while imposing an overhead of 2.32%. Regarding to the system's throughput, the service disruption was lower than 0.1% when a reconfiguration was performed. Another important result is the capability of finishing the reconfiguration when a CN reconnects. In this way, our proof of concept implementation took 31.50ms to complete the reconfiguration after the CN's regress. We expect to advance our work along the

following lines: (i) implementing and evaluating the dynamic dependency management, (ii) advancing in the field of reconfiguration policies and state reconfiguration, (iii) using a more realistic and complex evaluation application to better evaluate the proof of concept prototype, and (v) supporting legacy nodes that cannot perform a reconfiguration due to some limitation.

References

Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02*. . ACM Press. http://portal.acm.org/citation.cfm?doid=543613.543615, [accessed on Feb 11].

Chandy, K. M. and Lamport, L. (1985). Distributed Snapshots: Determining Global States of Distributed Systems. *ACM Trans. Comput. Syst.*, v. 3, n. 1, p. 63–75.

Chen, H., Yu, J., Hang, C., Zang, B. and Yew, P.-C. (sep 2011). Dynamic Software Updating Using a Relaxed Consistency Model. *IEEE Transactions on Software Engineering*, v. 37, n. 5, p. 679–694.

Cherniack, M., Balakrishnan, H., Balazinska, M., et al. (2003). Scalable distributed stream processing. In *In CIDR*.

Cugola, G. and Margara, A. (1 jun 2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, v. 44, n. 3, p. 1–62.

Eisenman, S. B., Miluzzo, E., Lane, N. D., et al. (2007). The BikeNet mobile sensing system for cyclist experience mapping. In *Proceedings of the 5th international conference on Embedded networked sensor systems - SenSys '07*. . ACM Press. http://portal.acm.org/citation.cfm?doid=1322263.1322273.

Ertel, S. and Felber, P. (2014). A framework for the dynamic evolution of highly-available dataflow programs. In *Proceedings of the 15th International Middleware Conference on - Middleware '14*. . ACM Press. http://dl.acm.org/citation.cfm?doid=2663165.2663320.

Ganti, R., Ye, F. and Lei, H. (nov 2011). Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, v. 49, n. 11, p. 32–39.

Ghafari, M., Jamshidi, P., Shahbazi, S. and Haghighi, H. (2012). An architectural approach to ensure globally consistent dynamic reconfiguration of component-based systems. In *Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering - CBSE '12*. . ACM Press. http://dl.acm.org/citation.cfm?doid=2304736.2304765.

Giuffrida, C., Iorgulescu, C. and Tanenbaum, A. S. (2014). Mutable Checkpoint-restart: Automating Live Update for Generic Server Programs. In *Proceedings of the 15th International Middleware Conference*. , Middleware '14. ACM. http://doi.acm.org/10.1145/2663165.2663328.

Hayden, C. M., Smith, E. K., Denchev, M., Hicks, M. and Foster, J. S. (2012). Kitsune: Efficient, General-purpose Dynamic Software Updating for C. In *Proceedings of the*

ACM International Conference on Object Oriented Programming Systems Languages and Applications. , OOPSLA '12. ACM. http://doi.acm.org/10.1145/2384616.2384635.

Kakousis, K., Paspallis, N. and Papadopoulos, G. A. (nov 2010). A survey of software adaptation in mobile and ubiquitous computing. *Enterprise Information Systems*, v. 4, n. 4, p. 355–389.

Kramer, J. and Magee, J. (1990). The evolving philosophers problem: dynamic change management. *IEEE Transactions on Software Engineering*, v. 16, n. 11, p. 1293–1306.

Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM*, v. 21, n. 7, p. 558–565.

Ma, X., Baresi, L., Ghezzi, C., Panzica La Manna, V. and Lu, J. (2011). Version-consistent dynamic reconfiguration of component-based distributed systems. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering - ESEC/FSE '11*. . ACM Press. http://dl.acm.org/citation.cfm?doid=2025113.2025148.

Morrison, J. P. (2010). *Flow-Based Programming, 2nd Edition: A New Approach to Application Development*. Paramount, CA: CreateSpace.

Nouali-Taboudjemat, N., Chehbour, F. and Drias, H. (2010). On Performance Evaluation and Design of Atomic Commit Protocols for Mobile Transactions. *Distrib. Parallel Databases*, v. 27, n. 1, p. 53–94.

Schweppe, H., Zimmermann, A. and Grill, D. (feb 2010). Flexible On-Board Stream Processing for Automotive Sensor Data. *IEEE Transactions on Industrial Informatics*, v. 6, n. 1, p. 81–92.

Silva, L. D., Vasconcelos, R., Lucas Alves, R. A., et al. (2012). A Large-scale Communication Middleware for Fleet Tracking and Management. In *Salão de Ferramentas, Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2012)*. . http://sbrc2012.dcc.ufmg.br/app/pdfs/p-06/SF-3-3.pdf.

Stonebraker, M., Çetintemel, U. and Zdonik, S. (1 dec 2005). The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, v. 34, n. 4, p. 42–47.

Vandewoude, Y., Ebraert, P., Berbers, Y. and D'Hondt, T. (dec 2007). Tranquility: A Low Disruptive Alternative to Quiescence for Ensuring Safe Dynamic Updates. *IEEE Transactions on Software Engineering*, v. 33, n. 12, p. 856–868.

Vasconcelos, R. O., Endler, M., Gomes, B. and Silva, F. (2013). Autonomous Load Balancing of Data Stream Processing and Mobile Communications in Scalable Data Distribution Systems. *International Journal On Advances in Intelligent Systems (IARIA)*, v. 6, n. 3&4, p. 300–317.

Vasconcelos, R. O., Vasconcelos, I. and Endler, M. (2014). A Middleware for Managing Dynamic Software Adaptation. In *13th International Workshop on Adaptive and Reflective Middleware (ARM 2014), In conjunction with ACM/IFIP/USENIX ACM International Middleware Conference 2014*. . http://dl.acm.org/citation.cfm?id=2677022.

Replicação de Máquina de Estado Baseada em Prioridade com PRaft

Paulo R. Pinho[1], Luciana de Oliveira Rech[1], Lau Cheuk Lung[1], Miguel Correia[2], Lásaro Jonas Camargos[3]

[1]Departamento de Informática e Estatística, Universidade Fedral de Santa Catarina, Brasil

prdepinho@gmail.com, {luciana.rech,lau.lung}@ufsc.br

[2]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal.

miguel.p.correia@tecnico.ulisboa.pt

[3]Faculdade de Computação, Universidade Federal de Uberlândia, Brasil

lasaro@ufu.br

Abstract. *State machine replication is an approach to create fault tolerant distributed systems. The system remains corret while tolerating faults from some of its replicas. These replicas must execute the same sequence of requests, a problem that is solved by total order algorithsm like Raft. some services may need that requests have different priority levels, such that some be executed before others. In this work, we propose an algorithm, PRaft, that introduces the idea of priority based state machine replication to Raft, modifying it to deal with request priority.*

Resumo. *Replicação de máquina de estado é uma abordagem para criar serviços distribuídos tolerantes a faltas. O sistema se mantém correto enquanto tolera a falta de algumas de suas réplicas. Essas réplicas devem executar a mesma sequência de requisições, problema que é resolvido por algoritmos de ordem total como o Raft. Alguns serviços podem precisar que as requisições tenha diferentes níveis de prioridade, para que umas sejam executadas antes de outras. Neste trabalho propõe-se um algoritmo, PRaft, que introduz a ideia de replicação de máquina de estado baseado em prioridades ao Raft, modificando-o para considerar a prioridades das requisições.*

1. Introdução

Sistemas de computação frequentemente estão sujeitos a faltas, com as quais deve lidar para ser um sistema confiável, evitando que faltas levem a *erros*, e erros levem a *falhas*. Mascaramento é uma abordagem de tolerância a faltas que consiste em adicionar processos redundantes que substituam processos que venham a sofrer faltas.

Replicação de máquina de estado (SMR) é uma técnica bem conhecida de tolerância a faltas em serviços distribuídos [Schneider 1990]. Em SMR um serviço é implementado por um conjunto de *máquinas de estado*, ou *réplicas*. Cada máquina de estado implementa o mesmo serviço, e devem executar a mesma sequência de operações. As operações devem ser determinísticas, de modo que duas máquinas de estado que iniciem no mesmo estado e executem a mesma sequência de operações para que levem o sistema

ao mesmo estado final e cheguem aos mesmos valores de retorno. Mesmo que algumas réplicas se desviem desse comportamento, as suas faltas são mascaradas, isto é, escondidas do cliente.

Para que todos os processos executem a mesma sequência de requisições, elas devem ser ordenadas usando um *protocolo de difusão de ordem total*. No centro desse protocolo mora o *problema do consenso*, em que um grupo de processos deve concordar com um único valor dentre um conjunto de valores propostos. A ordem total das requisições pode ser alcançada encadeando múltiplas instâncias de consenso, cada uma para decidir a ordem de uma requisição [Hadzilacos and Toueg 1993]. Já foi, contudo, demonstrado que não há algoritmo determinístico que resolva o problema de consenso sob a premissa de comunicação assíncrona [Fischer et al. 1985]. Para contornar esse obstáculo algoritmos geralmente estendem o requisito de tempo. Os algoritmos resolvem o problema de consenso garantindo sempre as propriedades de segurança(*safety*), mas garantem as propriedades de terminação apenas em casos específicos, previstos pelo novo requisito de tempo. Uma extensão comum é assumir sincronia mais fraca, como pela existência de um ponto no tempo a partir do qual faltas possam ser detectadas com confiança [Chandra and Toueg 1996]. Propuseram-se muitos algoritmos para resolver esse problema em ambos o modelos de faltas de *crash* [Lamport 2001, Lamport 1998, Oki and Liskov 1988, Ongaro and Ousterhout 2014] e bizantinas [Castro and Liskov 1999, Kotla et al. 2007]. Um algoritmo de consenso é o Raft [Ongaro and Ousterhout 2014], que garante segurança (*safety*) enquanto tolera f faltas em um sistema com $2f + 1$ processos.

Sistemas confiáveis podem ter requerimentos de tempo real, isto é, precisam que operações sejam executadas antes de prazos rígidos [Verissimo and Rodrigues 2001]. Atender tal requsito requer o uso de infraestruturas especializadas que permitam previsibilidade de comunicação e de execução. Isso é dificilmente o caso em sistemas distribuídos de uso geral, nos quais a comunicação inconfiável pode causar atrasos ou perdas de mensagens. Contudo é possível nesse caso atingir requerimentos menos estritos de tempo real – *soft* – [Locke 1986].

Muitos serviços podem tirar proveito de garantias de tempo real *soft* expressas em termos da diferença de níveis de urgência, ou prioridade. Por exemplo, diferentes prioridades podem ser atribuídas a usuários com contratos de serviço distintos em serviços de armazenamento na nuvem [Buyya et al. 2008, Bessani et al. 2011]. Alguns exemplos de serviços que se tornaram confiáveis usando SMR e que poderiam tirar proveito de prioridades são sistema de arquivos em rede [Castro and Liskov 1999, Kotla et al. 2007], serviços de *backup* cooperativo [Aiyer et al. 2005], e sistemas de gerência de bancos de dados relacionais [Luiz et al. 2011].

Há alguns trabalhos sobre algoritmos de difusão de ordem total baseada em prioridades [Nakamura and Takizawa 1992, Rodrigues et al. 1995, Wang et al. 2002]. O assunto apareceu primeiro em [Nakamura and Takizawa 1992], mas o algoritmo apresentado é sobre o modelo síncrono, e não é tolerante a faltas. Os outros trabalhos consideram o modelo de tempo assíncrono, mas precisam de algum *framework* baseado em detectores de faltas. O algoritmo de [Rodrigues et al. 1995] usa *view atomic multicast* [Schiper and Ricciardi 1993], e o algoritmo de [Wang et al. 2002] conta com o *general agreement framework* [Hurfin et al. 1999]. Essas abordagens resultam em um considerá-

vel número de mensagens trocadas entre as réplicas, e como resultado, têm complexidade de mensagens de $O(n^3)$ e $O(n^2)$ respectivamente.

Este artigo trata da extensão de SMR com prioridades, para que requisições de mais alta prioridade sejam executadas antes que as outras. Este problema é chamado de *replicação de máquina de estado baseado em prioridade* –PB-SMR. O problema SMR regular pode ser resolvido empregando um protocolo de ordem total como Raft para entregar a mesma sequência de requisições a todos os processos. Resta fazer com que a sequência seja ordenada segundo a prioridade das requisições.

É proposta neste artigo a modificação do algoritmo Raft [Ongaro and Ousterhout 2014] para lidar com a prioridade das requisições, sem qualquer premissa mais forte do que *eventual synchrony* [Dwork et al. 1988]. O protocolo resultante chama-se PRaft e resolve o PB-SMR dentro da complexidade linear de mensagens.

2. Trabalhos Relacionados

Não há muitos trabalhos de consenso baseado em prioridade na literatura. O pioneiro parece ser [Nakamura and Takizawa 1992], que propõe dois algoritmos. Um deles (PriTO) é uma difusão de ordem total baseado em prioridade no qual mensagens são entregues segundo as suas prioridades. Esse algoritmo depende, contudo, de um modelo de tempo síncrono, e não é tolerante a faltas. Diferentemente, a abordagem deste trabalho considera o modelo de tempo *eventual synchronous*, onde há um tempo de estabilização global depois do qual o sistema se comporta de maneira síncrona.

O artigo de [Rodrigues et al. 1995] propõe um algoritmo tolerante a faltas. Nesse protocolo, requisições de baixa prioridade são adicionadas no final da sequência de cada réplica, assumindo que as requisições são ordenadas a partir do início, onde ficam as de alta prioridade, para baixo, onde ficam as de baixa prioridade. Requisições de alta prioridade são inseridas num ponto no meio da sequência, baseado no progresso da réplica mais rápida. Antes de executar a nova requisição de alta prioridade, todas as réplicas precisam executar todas as requisições de mais baixa prioridade que a réplica mais rápida já executou. Por isso, o algoritmo não interrompe requisições de baixa prioridade para executar um requisição pronta de alta prioridade. O algoritmo assume a existência de um serviço de comunicação de grupo *virtual synchrony* que toma conta da transmissão de mensagens, gerenciamento de visão, tolerância a faltas e difusão atômica [Schiper and Ricciardi 1993]. A falta da capacidade de interrupção resulta em requisições de alta prioridade terem que esperar a execução de requisições de baixa prioridade antes de serem executadas. Essa é a principal diferença com este trabalho, já que o PRaft evita a inversão de prioridade interrompendo e reordenando requisições não confirmadas.

O trabalho de [Wang et al. 2002] propõe um algoritmo tolerante a faltas depende de *general agreement framework* de [Hurfin et al. 1999], que usa um detector de falta $\diamond S$ [Chandra and Toueg 1996] para resolver consenso. Cada vez que o sistema recebe uma requisição r, o serviço de consenso é invocado para definir o número de sequência da nova requisição. Requisições que são de prioridade mais baixa que r são movidas para a posição seguinte, para que r ocupe a sua posição na sequência. Se a réplica já havia começado a executar quaisquer requisições entre as que haviam sido movidas, então a réplica interrompe execução e restaura o estado de antes do processamento dessas requisições.

Quando a maioria das réplicas termina o processamento de r, r está confirmado e numa posição irreversível, donde outra requisição de prioridade mais alta não pode tirá-lo.

A principal diferença entre [Wang et al. 2002] é PRaft, e a principal motivação deste trabalho, é não usar *generic agreement framework* que faz as réplicas trocar mais mensagens que em PRaft. Em [Wang et al. 2002], dada uma lista de requisições para executar, o *framework* do algoritmo de consenso é invocado uma vez e mais $f + 1$ vezes, pelo menos, para cada requisição que confirma. Em PRaft, o acordo é invocado apenas uma vez para cada requisição, e mais uma vez quando ela é confirmada. Isso é alcançado lidando com a prioridade das requisições junto com o protocolo de consenso.

3. Modelo de Problema

O sistema é composto por um conjunto de Π processos, dos quais $f < |\Pi|/2$ podem faltar. Cada processo p mantém uma sequência *log* de requisições, um *termo* e um índice da requisição em processamento e. Uma requisição r no índice i do *log* do processo p é representado por $log_p[i]$.

Prioridade é denotada $P(r)$, e quando uma requisição r tem prioridade maior que r', então $P(r) > P(r')$.

Assume-se que um modelo de faltas em que os processos que faltam não se recuperam, chamado de faltas de *crash*.

Comunicação entre réplicas é feito apenas com troca de mensagens. Mensagens podem se perder, são entregues no máximo uma vez e não estão sujeitas a corrupção.

Considera-se o modelo assíncrono ampliado com o tempo global de estabilização (GST), chamado *eventual synchrony* [Cristian and Fetzer 1998, Fetzer and Cristian 1995]. O sistema comporta-se de maneira assíncrona, em que mensagens podem levar tempo arbitrário de transmissão até a entrega. O sistema mais cedo ou mais tarde entra em um período estável em que as mensagens são transmitidas e as requisições são executadas com sincronia. Esse modelo é necessário para garantir que o algoritmo termine. Períodos de estabilização duram suficiente tempo para o algoritmo terminar, o que acontece com frequência em sistemas reais.

Assume-se que o tempo de execução das requisições seja significativamente longo em comparação com o tempo de execução do protocolo. Se requisições de baixa prioridade são longas, esperar pelo seu término pode ser contra-produtivo para o sistema quando requisições de alta prioridade que demandam atenção imediata se mantêm atrasadas.

Para definir o problema PB-SMR, deve-se definir Inversão Local de Prioridade (LPI) e Inversão Global de Prioridade (GPI), segundo [Wang et al. 2002]. Uma requisição r está invertida no processo p ($LPI_p(r)$) se r está preparado para execução p enquanto executa outra requisição de menor prioridade que r. Uma requisição r está invertida globalmente ($GPI(r)$) quando, para cada processo p que terminou a execução de r, $LPI_p(r)$. Para que não ocorra $GPI(r)$, é necessário que pelo menos um processo tenha executado r sem inversão.

O algoritmo PRaft satisfaz as seguintes propriedades:

- Não trivialidade: Se uma requisição r foi confirmada por qualquer processo, então r foi proposto.

- Acordo: Se quaisquer dois processos p e q confirmaram $log[i]$, então $log_p[i] = log_q[i]$.
- Ordem de prioridade: Requisições não confirmadas são confirmadas segundo a sua prioridade.

4. Raft

Raft é um algoritmo de consenso desenvolvido para ser simples de fácil entendimento. Paxos [Lamport 1998], apesar de ser um algoritmo de consenso mais conhecido, é notável pela sua complexidade. Compreendê-lo é um desafio para muitas pessoas, mesmo pesquisadores experientes [Ongaro and Ousterhout 2014].

Raft tem como objetivo fazer com que os processos concordem em executar a mesma sequência de requisições, enquanto toleram o *crash* de uma parcela de processos. A relação entre o total de processos n e o total de faltas f é dada pela equação: $n = 2f+1$. Uma requisição está confirmada quando a maioria dos processos concordou em executá-la. Em cada processo há uma máquina de estado ao qual ele aplica requisições para processamento. Para uma requisição ser aplicada à máquina de estado ela precisa estar confirmada e todas as requisições anteriores devem estar processadas. Um processo envia a mensagem de retorno de uma requisição ao cliente quando a máquina de estado termina o seu processamento.

Raft é constituído de um conjunto de processos, cada um dos quais está em um dos estados *líder*, *seguidor* ou *candidato*. Normalmente um dos processos é líder, e os demais os seus seguidores. As mensagens vão sempre do líder para os seguidores e dos seguidores respondendo ao líder, exceto na eleição, como logo será explicado. Há dois tipos de mensagens trocadas pelos processos: *AppendEntry*, usado pelo líder para replicar o seu *log*, e *RequestVote*, usado pelo candidato de uma eleição para pedir votos.

O tempo é dividido em *termos*. Um termo é o período que começa com a eleição de um candidato e termina com a falta do líder eleito. É representado por um número, e todos os termos sucessivos são maiores que os anteriores. O termo atual é sempre enviado junto a qualquer mensagem transmitida no sistema. Quando um processo recebe uma mensagem com o termo menor que o seu, ele a ignora. Quando o processo recebe uma mensagem de termo maior, então ele atualiza o seu termo para ficar igual e torna-se seguidor do líder vigente. No máximo apenas um líder opera em cada termo.

A Figura 1 ilustra a sequência de passos do algoritmo desde a eleição até a confirmação de requisições. Essa sequência é composta por três passos. O primeiro passo é a eleição do líder (*RequestVote* e *Vote*), que acontece uma vez no início de cada termo. Eleito o líder, ele passa a receber requisições de clientes replicando-as no segundo passo (*AppendEntry* e Resposta). O terceiro passo é a confirmação das requisições aos seguidores (*AppendEntry*), mensagem que pode coincidir com a replicação de requisições que chegaram depois do *AppendEntry* anterior. Esses passos serão explicados a seguir.

Todos os processos iniciam no estado *seguidor*, que espera receber periodicamente mensagens *AppendEntry* do líder para se manter no seu mandato. Se o seguidor não recebe *AppendEntry* dentro de um *timeout*, ele assume que o líder falhou e inicia uma eleição candidatando-se para ser o próximo líder.

O processo muda o seu estado para *candidato*, incrementa o seu termo e envia

uma mensagem *RequestVote* com o termo atual e o índice da última entrada do seu *log*. Um processo que recebe essa mensagem responde se o seu termo for menor que o do candidato e o seu *log* não for mais atualizado que o dele. Um log_p é mais atualizado que um log_q se o termo da última entrada de log_p for maior que o termo da última entrada de log_q, ou, se forem iguais, se o índice da sua última entrada for maior.

Um candidato que recebe o voto da maioria dos processos muda o seu estado para *líder*, passa a receber requisições de clientes e a enviar mensagens *AppendEntry* periodicamente para replicar requisições e manter o seu mandato.

Se dois ou mais processos se candidataram mais ou menos ao mesmo tempo e os votos ficaram divididos entre eles, pode acontecer de nenhum deles ganhar a maioria dos votos. Se um candidato não for eleito dentro de um *timeout*, ele incrementa o seu termo e elege-se de novo. Esses *timeouts* são aleatórios dentro de uma faixa de milissegundos, para impedir que múltiplos processos candidatem-se sempre simultaneamente.

O *log* de um processo é uma lista de entradas na forma (r, t), onde r é a requisição, e t é o termo em que a entrada foi incluída no log. O líder, ao receber requisições de clientes, anexa as requisições no final do log.

A mensagem *AppendEntry* que o líder envia periodicamente não serve só de *keep-alive* para os seus seguidores, mas também para replicar as requisições que o líder acumula no seu *log* entre um *AppendEntry* e outro, enviando a cada seguidor a parte do *log* que lhe falta. Quando o seguidor s responde ao líder informando que o *log* foi replicado com sucesso, o líder atualiza o $match\_index_s$, que indica até onde o *log* dos dois estão iguais. Quando o $match\_index$ de f seguidores tiver atingido um índice i, então atualiza-se o $commit\_index = i$. Até o índice i as requisições estão confirmadas e o líder pode aplicá-las na máquina de estado e enviar as respostas da execução ao cliente requisitante.

Para o seguidor s replicar as requisições com sucesso, o seu log_s deve estar igual ao do líder até o índice da próxima requisição $next\_index_s$. O seguidor compara o $next\_index_s$ enviado pelo líder com o termo da entrada no índice $next\_index_s - 1$. Se forem iguais, significa que os $logs$ dos dois são iguais até aquele ponto. Se forem diferentes, o seguidor responde ao líder que a replicação foi mal sucedida. Então o líder decrementa $next\_index_s$ e tenta de novo no próximo *AppendEntry*. Cada vez que $next\_index_s$ diminui, maior é a sublista de requisições enviadas ao seguidor no *AppendEntry*, e isso se repete até chegar o ponto em que o log dos dois são iguais e a replicação é bem sucedida.

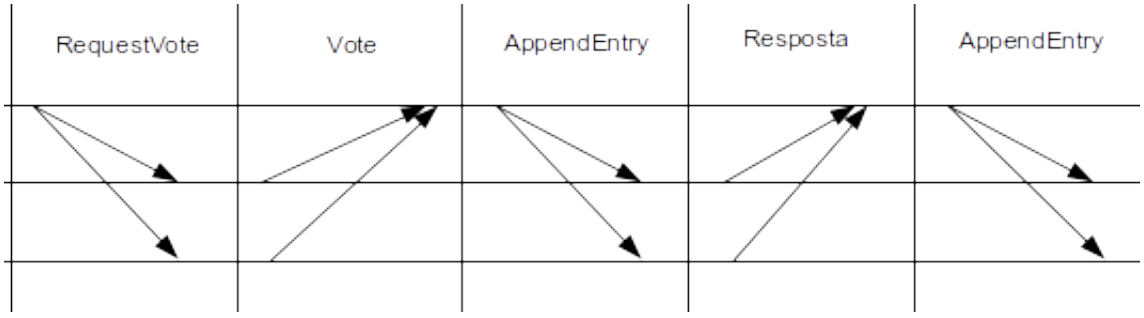


Figura 1. Sequência de passos do Raft

O algoritmo garante que o líder sempre terá todas as requisições confirmadas antes da sua eleição, já que pelo menos um processo com o *log* inteiro sempre está disponível, e o líder eleito é sempre o que tem o *log* mais atualizado.

5. PRaft

As principais contribuições do PRaft (Priority Raft) são a execução antecipada das requisições, a ordem de prioridade e a interrupção de execução. No PRaft os processos não esperam a requisição estar confirmada para executá-la, mas executam-nas na hora em que as recebem. A resposta ao *AppendEntry* não é usado para confirmar as entradas, mas apenas para informar que o *log* foi replicado. A variável $match\_index$ é trocada pela $execution\_index$, pois o critério para a confirmação das entradas não é mais a correspondência entre os *logs* dos seguidores com o do líder, mas com as requisições que os processos tenham terminado de executar.

Os processos mantêm o progresso por meio de dois índices. O progresso de execução é indicado pelo índice e, que marca qual requisição a máquina de estado está executando no momento. E o progresso de confirmação é indicado pelo $commit\_index$, que é incrementado toda vez que a maioria dos processos termina a execução de uma requisição.

Cada requisição r tem uma prioridade definida pelo cliente denominada $P(r)$. O líder sempre ordena as requisições não confirmadas segundo a prioridade delas.

Os valores iniciais das variáveis são a presentadas no Algoritmo 1

O Algoritmo 2 mostra quando o líder l recebe uma requisição r de um cliente c. Calcula-se o valor da posição i que r deve assumir no *log*. Todas as requisições não confirmadas que estiverem em posição maior ou igual a i devem dar espaço para r. Ocorre uma *inversão de prioridade* (linha 2) quando uma dessas requisições é a que está em execução na máquina de estado. Quando isso acontece, a máquina de estado interrompe a execução e restaura o estado até o ponto anterior à posição i. O líder inclui r no *log*, que assume o índice i, e move ao índice seguinte os processos de prioridade menor que r. Então aplica $log[i]$ à máquina de estado, que volta à execução habitual.

Algorithm 1 Valores iniciais

1: $commit\_index \leftarrow -1$;
2: $e \leftarrow \infty$;
3: $next\_index_s \leftarrow 0$;
4: $execution\_index_p \leftarrow -1$.

O Algoritmo 3 descreve o método para definir a posição i da nova requisição r. Novas requisições nunca ocupam posições de requisições confirmadas. Por isso a posição a ser ocupada jamais será menor ou igual a $commit\_index$. A posição é definida levando em consideração a prioridade de r e das requisições não confirmadas do *log*.

Se a requisição r não causou inversão de prioridade, e como consequência a máquina de estado não interrompeu a execução e não restaurou o estado anterior a i, então a máquina de estado segue a execução de $log[e]$ normalmente.

Como consequência da reordenação do *log*, o $next\_index$ de todos os seguidores (linha 11) e $match\_index$ de todos os processos (linha 16) devem ser atualizados para

Algorithm 2 Líder l recebe requisição r de cliente.

```
1: i ← posição em que r deve ser inserido no log
2: if e ≥ i then
3:     State machine interrompe execução.
4:     State machine restaura o estado snapshot[i − 1]
5: end if
6: Insere r no log
7: if State machine estiver inativa then
8:     e ← i
9:     Aplica log[e] à State machine
10: end if
11: for Para cada seguidor s do
12:     if next_index_s > i then
13:         next_index_s ← i
14:     end if
15: end for
16: for Para cada processo p do
17:     if executed_index_p ≥ i then
18:         executed_index_p ← i − 1
19:     end if
20: end for
```

Algorithm 3 Determina a posição em que a requisição r é inserida.

```
1: for i ← len(log) − 1; i > commit_index; i ← i − 1 do
2:     if P(log[i]) ≥ P(r) then
3:         return i + 1
4:     end if
5: end for
6: return commit_index + 1
```

que a seção modificada do *log* seja replicada para todos os seguidores e a execução de alguma requisição dessa seção seja desconsiderada.

O Algoritmo 4 mostra os parâmetros do *AppendEntry* que o líder envia periodicamente aos seus seguidores. A variável $entries$ é a sub-lista das entradas que faltam à réplica, todas as entradas do $next\_index_p$ em diante. O seguidor não deve aceitar as novas entradas se o seu *log* não estiver igual ao do líder até o ponto do $next\_index$. São enviados os parâmetros $prev\_index$ e $prev\_term$ para o seguidor fazer a comparação. São enviados também o termo, que vai em todas as mensagens do protocolo, e o $commit\_index$ para informar o seguidor quais requisições estão confirmadas.

O Algoritmo 5 descreve o seguidor quando recebe o *AppendEntry*. O seguidor aceita o *AppendEntry* se seu *log* for igual ao do líder até aquele ponto (linha 1). Como no Raft, o *log* do seguidor deve se conformar com o do líder em toda a seção de índice $prev\_index+1$ em diante. Ocorre inversão de prioridade se o seguidor estiver executando alguma requisição dessa secção (linha 2). Nesse caso, o seguidor faz a mesma coisa que o líder teria feito numa inversão de prioridade. Então o seguidor atualiza o seu *log* incluindo

Algorithm 4 Líder l envia AppendEntry aos seguidores

```
1: for para cada seguidor s do
2:     entries ← log[next_index_s..]
3:     prev_index ← next_index_s − 1
4:     prev_term ← log[prev_index].term
5:     Envia um AppendEntry(term, entries, commit_index, prev_index, prev_term)
6: end for
```

Algorithm 5 Seguidor s recebe $AppendEntry\ AE$ do líder l

```
1: if log_s[prev_index_AE].term = prev_term_AE ∨ prev_index_AE = −1 then
2:     if e ≥ prev_index_AE + 1 then
3:         State machine interrompe execução.
4:         State machine restaura o estado snapshot[prev_index_AE]
5:     end if
6:     Atualiza log
7:     Atualiza commit index
8:     if State machine estiver inactiva then
9:         e ← prev_index_AE + 1
10:        Aplica log[e] à State machine
11:    end if
12:    Responde ao líder "Bem sucedido"
13: else
14:    Responde ao líder "Mal sucedido"
15: end if
```

todas as requisições que vieram no *AppendEntry*, substituindo as requisições que estavam nesses índices pelas novas. O seguidor então envia uma resposta ao líder indicando que a replicação foi bem sucedida (linha 12).

Se o *log* do seguidor está diferente do *log* do líder até o ponto $prev\_index$, então o seguidor responde com uma mensagem de fracasso (linha 14).

O Algoritmo 6 descreve quando o líder recebe uma resposta do seguidor s. Se a mensagem de operação for bem sucedida, $next\_index_s$ é atualizado (linha 2) para que o líder não envie as mesma requisições a s nos próximos *AppendEntry*. Se a mensagem for de operação mal sucedida (linha 4), então o líder decrementa $next\_index_s$ para enviar uma sublista maior no próximo *AppendEntry*. À medida que o seguidor continua respondendo que a operação foi mal sucedida, o índice vai decrementando e a sublista enviada ao seguidor vai aumentando até achar o ponto em que o *log* dele esteja igual ao do líder, ou ao início, quando o *log* inteiro é replicado.

Os Algoritmos 7 e 8 descrevem o que acontece quando uma máquina de estado termina a execução de uma requisição. Em qualquer processo, seja líder ou seguidor, a máquina de estado está sempre executando requisições, que são aplicadas uma a uma, na ordem em que estão disposta no *log*. Quando a máquina de estado termina a execução de uma requisição, o processo aplica a requisição seguinte na máquina de estado e, se for seguidor, envia uma mensagem ao líder. No Algoritmo 7 o seguidor envia nessa mensagem a variável $exec$, isto é, o índice da requisição que ele acabara de executar

Algorithm 6 Líder l recebe resposta res do seguidor s para o AppendEntry

1: **if** a resposta for "Bem sucedido" **then**
2: $next\_index_s \leftarrow len(log)$
3: **else**
4: $next\_index_s \leftarrow next\_index_s - 1$
5: **end if**

(linha 7). O líder, tendo terminado o processamento ou tendo recebido uma mensagem de término de um seguidor, atualiza $execution\_index_s$ e incrementa o $commit\_index$ se possível(linha 3).

Algorithm 7 State machine de um processo termina a execução da requisição $log[e]$

1: $e \leftarrow e + 1$
2: State machine aplica $log[e]$
3: **if** é o líder **then**
4: $execution\_index_l \leftarrow e - 1$
5: Atualiza o $commit\_index$
6: **else**
7: Envia ao líder a mensagem de fim de execução da requisição em $exec \leftarrow e - 1$.
8: **end if**

Algorithm 8 O líder recebe do seguidor s a mensagem msg de fim de execução

1: **if** $exec_{msg} < next\_index_s$ **then**
2: $execution\_index_s \leftarrow exec_{msg}$
3: Atualiza o $commit\_index$
4: **end if**

O Algoritmo 9 descreve como $commit\_index$ é incrementado. Quando a maioria dos processos tiver executado até o índice i, o líder terá $executed\_index_s \geq i$ para cada processo s que faz parte da maioria dos processos. O $commit\_index$ é o menor valor dentre os $f + 1$ maiores $executed\_index$. Assim todas as requisições que foram executadas pela maioria dos processos são confirmadas.

Algorithm 9 Atualiza $commit\_index$

1: $buf \leftarrow$ a lista com a $execution\_index_s$ de todos os processos
2: Dispõe buf em ordem decrescente
3: $index \leftarrow buf[f]$
4: **if** $index > commit\_index$ **then**
5: $commit\_index \leftarrow index$
6: **end if**

6. Avaliação

Esta seção avalia PRaft e o compara com os trabalhos relacionados. A avaliação de protocolos distribuídos normalmente é feita em termos de duas métricas: o número de passos de comunicação e o número de mensagens enviadas.

Avaliou-se neste trabalho o caso normal de processamento do protocolo, sem faltas nem troca de líder. PRaft segue o mesmo número de passos de comunicação que o Raft, mas tem apenas $n-1$ mensagens a mais. As mensagens contadas são $n-1$ *AppendEntry*s, $n-1$ mensagens de resposta ao *AppendEntry*, $n-1$ mensagens de término de processamento, e o $n-1$ *AppendEntry* seguinte que informa os seguidores do $commit\_index$ atualizado. As mensagens de votação não são contadas pois são feitas apenas uma vez por termo. As mensagens mais recorrentes são as três referidas, que somam $4(n-1)$ mensagens, onde n é o total de processos, e a complexidade é linear $O(n)$.

A tabela 1 mostra a comparação entre o PRaft e os algoritmos estudados. PRaft tem a menor complexidade de mensagens entre eles. O algoritmo de [Rodrigues et al. 1995] usa *view atomic multicast* [Schiper and Ricciardi 1993] que executa com $(n-1)^2$ mensagens, e [Wang et al. 2002] usa *general agreement framework* [Hurfin et al. 1999] que executa trocando $n+2n^2$ mensagens. Ambos possuem complexidade cúbica, devido aos serviços de consenso que usam. PritTO, de [Nakamura and Takizawa 1992] tem complexidade apenas quadrática, mas foi feito para sistemas síncronos. PRaft, tendo sido feito à semelhança do Raft [Ongaro and Ousterhout 2014], executa em tempo linear.

A Figura 2 mostra a comparação do desempenho de execução entre o Raft e o PRaft. Foram feitos testes com os dois algoritmos recebendo requisições de 20 clientes, que enviavam 100 requisições cada. Todos os clientes enviam requisições simultaneamente, e cada um só envia a próxima requisição depois que recebe a resposta da anterior. A cada requisição é associada uma prioridade aleatória entre 0 e 10, o que garante, por causa da regularidade da distribuição aleatória dos valores, que há um número aproximadamente igual de requisições em cada prioridade. O tempo de execução é fixo em 1 segundo.

A Figura 2 ilustra a latência da execução do Raft, notado com triângulos, e do PRaft, notado com cruzes. Como o Raft não distingue prioridades d as requisições, a média do tempo de execução é próxima de 18.91 segundos para todas as prioridades, com desvio padrão de 0.95 segundos. Já o PRaft discrimina a prioridade das requisições.

Na Figura 3 se podem ver os diagramas de barra da média de latência do PRaxos. No gráfico (a) está claro que requisições de prioridade 0 têm latência média muito maior que as demais. Isso acontece porque os clientes enviam as suas requisições sem pausa ao serviço, e se um cliente envia uma requisição de prioridade 0 no início dos testes, ele terá que esperar o processamento das requisições dos outros clientes antes de receber a resposta da sua. A média da latência para as requisições de prioridade 0 é de 256.28

Tabela 1. Comparação dos algoritmos relacionados

Algorítmo	Modelo de Tempo	# passos	# mensagens	Complex.
PritTO	síncrono	3	$(n-1)^2$	$O(n^2)$
Rodrigues et al.	detector de faltas	2	$(n-1)+n(n-1)^2$	$O(n^3)$
Wang et al.	detector de faltas	4	$(2n^3+7n^2+5n-2)/2$	$O(n^3)$
Raft	assíncrono	3	$3(n-1)$	$O(n)$
PRaft	*eventual synchronous*	3	$4(n-1)$	$O(n)$

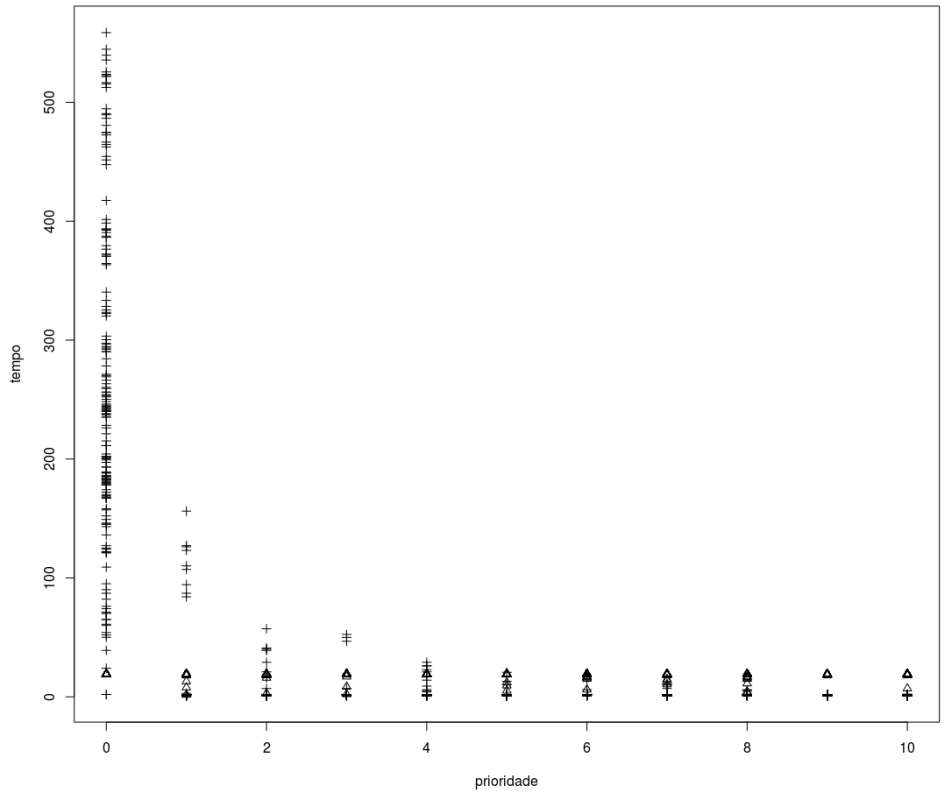
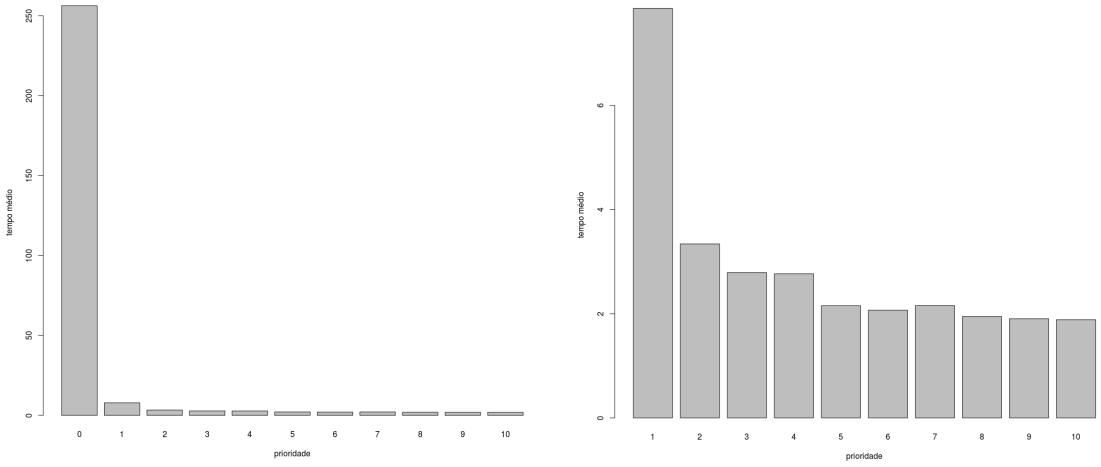


Figura 2. Desempenho comparativo entre a latência do Raft e do PRaft



(a) (b)

Figura 3. Diagramas de barra da média de latência do PRaft.

segundos, e alto desvio padrão de 134.63 segundos. O gráfico (b) mostra o diagrama de barras sem as requisições de prioridade 0 para facilitar a visualização. A latência das requisições diminui à medida em que aumenta a sua prioridade. A partir da prioridade 1, com latência média de 7.86 segundos, até as requisições de prioridade 10, que têm

latência média de 1.89 segundos, e desvio padrão de 0.31 segundos. Como se pode ver, a média da latência do Raft, 18.91 segundos, é maior do que a latência da requisição de prioridade 10 do PRaft, 1.89 segundos, isto é, dez vezes mais rápido. Isso demonstra que o PRaft é adequado para o processamento de requisições com prioridade.

7. Conclusão

Neste artigo foi proposto um algoritmo de consenso baseado em prioridade para a replicação de máquina de estado que não requer um serviço de consenso, e portanto funciona com menos mensagens que os algoritmos relacionados. Isso foi alcançado adaptando o algoritmo Raft para receber requisições e organizá-las segundo suas prioridades para que as de mais alta prioridade sejam executadas antes das de menor prioridades e evite inversão de prioridades. PRaft mantém as garantias de segurança (*safety*) e tolera $f = x/2$ faltas. As principais contribuições do PRaft são a execução antecipada das requisições, a ordem de prioridade, e a interrupção de execução. Esse protocolo pode ser utilizado na prática para suportar prioridades em serviços replicados como os de armazenamento em nuvem, serviços de arquivo em rede, *backup*s cooperativos e gerenciadores de banco de dados relacionais.

Referências

Aiyer, A. S., Alvisi, L., Clement, A., Dahlin, M., Martin, J., and Porth, C. (2005). BAR fault tolerance for cooperative services. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles*.

Bessani, A., Correia, M., Quaresma, B., André, F., and Sousa, P. (2011). DepSky: Dependable and secure storage in a cloud-of-clouds. In *Proceedings of the 6th ACM SIGOPS/EuroSys European Systems Conference*, pages 31–46, Saltzburg, Austria. ACM, New York, NY.

Buyya, R., Yeo, C. S., and Venugopal, S. (2008). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*, pages 5–13.

Castro, M. and Liskov, B. (1999). Practical Byzantine fault tolerance. In *Proceedings of the 3rd USENIX Symposium on Operating Systems Design and Implementation*, pages 173–186.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267.

Cristian, F. and Fetzer, C. (1998). The timed asynchronous system model. In *Proceedings of the 28th IEEE International Symposium on Fault-Tolerant Computing*, pages 140–149.

Dwork, C., Lynch, N., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323.

Fetzer, C. and Cristian, F. (1995). On the possibility of consensus in asynchronous systems. In *Proceedings of the 1995 Pacific Rim International Symposium on Fault-Tolerant Systems*, pages 86–91.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382.

Hadzilacos, V. and Toueg, S. (1993). Fault-tolerant broadcasts and related problems. In Mullender, S., editor, *Distributed Systems*, pages 97–145. ACM Press/Addison-Wesley.

Hurfin, M., Macedo, R. A., Raynal, M., and Tronel, F. (1999). A general framework to solve agreement problems. In *Proceeding of the 18th International Symposium on Reliable Distributed Systems*, pages 56–65.

Kotla, R., Alvisi, L., Dahlin, M., Clement, A., and Wong, E. (2007). Zyzzyva: speculative Byzantine fault tolerance. In *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*, pages 45–58.

Lamport, L. (1998). The part-time parliament. *ACM Transactions Computer Systems*, 16(2):133–169.

Lamport, L. (2001). Paxos made simple. *ACM Sigact News*, 32(4):18–25.

Locke, C. D. (1986). *Best-effort Decision-making for Real-time Scheduling*. PhD thesis.

Luiz, A. F., Lung, L. C., and Correia, M. (2011). Byzantine fault-tolerant transaction processing for replicated databases. In *Proceedings of the 10th IEEE International Symposium on Network Computing and Applications*, pages 83–90.

Nakamura, A. and Takizawa, M. (1992). Priority-based total and semi-total ordering broadcast protocols. In *Proceedings of the 12th International Conference on Distributed Computing Systems*, pages 178–185.

Oki, B. M. and Liskov, B. (1988). Viewstamped replication: A new primary copy method to support highly-available distributed systems. In *Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing*, pages 8–17.

Ongaro, D. and Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference*, pages 305–319.

Rodrigues, L., Veríssimo, P., and Casimiro, A. (1995). Priority-based totally ordered multicast. In *3rd IFIP/IFAC Workshop on Algorithms and Architectures for Real-Time Control*.

Schiper, A. and Ricciardi, A. (1993). Virtually-synchronous communication based on a weak failure suspector. In *Proceedings of the 23rd International Symposium on Fault-Tolerant Computing*, pages 534–543.

Schneider, F. B. (1990). Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

Verissimo, P. and Rodrigues, L. (2001). *Distributed Systems for System Architects*. Kluwer Academic Publishers.

Wang, Y., Anceaume, E., Brasileiro, F., Greve, F., and Hurfin, M. (2002). Solving the group priority inversion problem in a timed asynchronous system. *IEEE Transactions on Computers*, 51(8):900–915.

Redução de Tráfego de Jogos Distribuídos Através da Predição de Movimento Baseada em Aprendizado de Máquina

Pâmela de Assis Beltrani, Aurora T. R. Pozo, Elias P. Duarte Jr.

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

{pabeltrani,aurora,elias}@inf.ufpr.br

Abstract. *Distributed game players maintain a consistent vision of the positions of each other by periodically exchanging messages. Besides the fact that these messages can suffer delays that cause rendering inconsistencies, they also represent an overhead on the network. In order to avoid sending these messages, strategies for movement predicton, such as Dead Reckoning, can be employed. In this work, we present a movement prediction strategy based on machine learning. The strategy consists of two phases. In the first phase, a learning model classifies whether Dead Reckoning is able to predict the new position correctly or not. In case of a mistake, another learning model is used to predict the new direction. The proposed strategy has been applied to the World of Warcraft game. To evaluate the new strategy, the learning models were implemented with the Weka tool using real trace data. Results show that a traffic reduction of 66.41% on average, which is superior to other approaches.*

Resumo. *Em jogos distribuídos* multiplayer *os jogadores mantém uma visão consistente das posições uns dos outros através da troca periódica de mensagens. As mensagens de atualização, além de poderem sofrer atrasos de entrega que causam inconsistências na renderização, representam uma sobrecarga na rede. Estratégias de predição de movimento, como o tradicional algoritmo* Dead Reckoning*, evitam a troca de mensagens quando acertam. Neste trabalho, apresentamos uma nova estratégia para a predição de movimento baseada em aprendizado de máquina. A estratégia consiste de duas fases. Na primeira, um modelo de aprendizado classifica se o* Dead Reckoning *acerta ou erra. Em caso de erro é utilizado um outro modelo de aprendizado para prever a nova posição. A estratégia proposta foi aplicada para o jogo* World of Warcraft*. Para a avaliação, os modelos de aprendizado foram implementados na ferramenta Weka. Foram utilizados dados de* traces *reais do jogo. Resultados mostram que a redução de tráfego foi de 66,41% em média, superior a outras abordagens.*

1. Introdução

Os jogos eletrônicos estão atingindo níveis de popularidade extremamente elevados. A *Entretaiment Software Association* (ESA), aponta que em 2015 são 155 milhões os norte-americanos que jogam video-games freqüentemente [1]. A ESA também aponta que de cada cinco casas norte-americanas em quatro delas existe algum dispositivo eletrônico que é

[1]http://www.theesa.com/facts/pdfs/esa_ef_2014.pdf

utilizado para jogar. Essa presença cada vez mais marcante dos jogos na vida das pessoas se reflete diretamente na comercialização dos mesmos. Por exemplo, o jogo *Grand Theft Auto V* (GTA V), lançado em setembro de 2013, vendeu 11,21 milhões de cópias no dia de seu lançamento [2]. Outro caso de sucesso é o jogo *Call of Duty: Black Ops II* (CODII) que vendeu um total de 13,62 milhões de cópias [3]. Diversos destes jogos são distribuídos e *multiplayer*, permitindo que vários jogadores interajam através da Internet.

Em jogos distribuídos, os jogadores devem ter uma visão consistente do estado do mundo do jogo, de forma que múltiplos jogadores têm a mesma visualização. Para que essa visão seja consistente, as informações sobre o estado do jogo são transmitidas através da troca periódica de mensagens. Dentre as informações que precisam ser trocadas, estão as informações sobre a posição de cada avatar, a entidade que representa um jogador dentro do jogo, e a sua movimentação. Entretanto, estas mensagens de atualização de movimento dos avatares, além de poderem sofrer atrasos que causam inconsistências ou ainda saltos abruptos de renderização, representam uma sobrecarga na rede.

Tradicionalmente, o algoritmo *Dead Reckoning* é utilizado para prever a movimentação que os avatares realizam e, quando acerta, evita a troca de mensagens [Standards Committee on Interactive Simulation 1995]. Por exemplo, considere que um cliente deve prever a posição de um avatar de um outro jogador; em ambos os clientes o *Dead Reckoning* é executado e somente ocorre uma troca de mensagens quando é constatado que a previsão está incorreta. Para fazer essa previsão, o *Dead Reckoning* utiliza as leis da física e assume que esta movimentação será em linha reta. Em outras palavras, o *Dead Reckoning* não considera que o avatar poderá se movimentar em uma nova direção. Como em jogos é desejavel que o jogador tenha muita interação, e é comum que essa interação seja através da movimentação seu avatar, é notória a baixa taxa de precisão do *Dead Reckoning* nestes ambientes. Uma abordagem baseada em computação bio-inspirada para melhorar a taxa de precisão do *Dead Reckoning* é o algoritmo *AntReckoning* [Yahyavi et al. 2012]. O problema desta abordagem é que necessita que um especialista no jogo configure os multiplos parâmetros utilizados pelo *AntReckoning* o que dificulta a utilização desta abordagem na prática.

Neste trabalho, apresentamos uma nova estratégia para a predição da movimentação de avatares em jogos distribuídos baseada em aprendizado de máquina. Esta estratégia tem duas fases bem definidas. Na primeira fase, um modelo de aprendizado classifica se o *Dead Reckoning* acerta ou erra a predição de um determinado avatar. Em caso de acerto, o *Dead Reckoning* é utilizado pra fazer a previsão. Porém, em caso de erro, a segunda fase é executada. Nessa segunda fase, é utilizado um novo modelo de aprendizado para fazer a previsão da nova direção de movimento. Os modelos de aprendizado são construídos utilizando os algoritmos: *Local Weighted Learning* [Atkeson et al. 1997], *Bootstrap Agregating* (Bagging) [Ripley 1996], *Multilayer Perceptron* [Quinlan 1987] e *Reduced Error Pruning Tree* [Opitz and Maclin 1999].

A estratégia proposta foi aplicada para o jogo *World of Warcraft*[4] . Os modelos de aprendizado foram construídos utilizando uma nova base de dados de traces do jogo.

[2]http://www.guinnessworldrecords.com/news/2013/10/confirmed-grand-theft-auto-breaks-six-sales-world-records-51900/

[3]http://www.vgchartz.com/game/70716/call-of-duty-black-ops-ii/

[4]World of Warcraft: http://us.battle.net/wow/pt/

Esta nova base de dados foi construída utilizando dados de *traces* que estão disponíveis publicamente [Shen et al. 2014]. Esta base contém informações sobre o campo de visão de cada um dos jogadores e apresenta informações sobre quais jogadores e NPCs (*Non-Playable Characters*) estão visíveis por determinado jogador em um instante de tempo.

Resultados mostram que a estratégia proposta obtém uma taxa de acerto médio de 76,60% para a primeira fase; e de 51,02% para a segunda fase. Destaca-se o algoritmo *Bagging*, que obtém uma taxa de acerto para a primeira e a segunda fases de 81,10% e 73,37%, respectivamente. Os testes executados mostram também uma taxa de acerto final médio de 66,41%, observa-se que nesta mesma base o algoritmo *Dead Reckoning* obteve a taxa de acerto de 57,89%. Destaca-se novamente o algoritmo *Bagging* obteve uma taxa de acerto final de 78,76%. Estes resultados mostram que a utilização de aprendizado de máquina para a predição da movimentação de avatares jogos distribuídos é uma alternativa muito atraente. Em comparação com o AntReckoning, a sua a taxa de sucesso é semelhante, cerca de 30% de melhoria sobre o *Dead Reckoning*. Entretanto, a grande vantagem da estratégia proposta é que não necessita de um especialista para setar os múltiplos parâmetros utilizados por aquele modelo. Utilizando aprendizado de máquina, os parâmetros são "automaticamente aprendidos".

O restante deste trabalho está organizado da seguinte forma. Na Seção 2 são apresentados conceitos básicos de jogos distribuídos. Em seguida, na Seção 3, os algoritmos *Dead Reckoning* e o *AntReckoning* são apresentados. Na Seção 4 é apresentada uma breve introdução aos conceitos de aprendizado de máquina. Na Seção 5 a estratégia proposta é descrita. Os resultados obtidos para o jogo *World of Warcraft* são apresentados na Seção 6. Por fim, a conclusão segue na Seção 7.

2. Jogos: Definições Básicas

Nessa seção são apresentados, de forma sucinta, os principais os conceitos de jogos que são necessários para a compreensão da contribuição deste trabalho. Nos jogos baseados em modelo de avatar o jogador interage com o jogo por meio de um personagem representativo, o avatar, que existe em uma localização do mundo do jogo [Claypool and Claypool 2010]. Pelo avatar o jogador consegue apenas ver e interagir com o que está presente em suas imediações. Nos jogos com avatar pode existir uma diferença de perspectiva, esta pode ser tanto em primeira quanto em terceira pessoa. Existem alguns jogos que permitem inclusive a escolha da visão utilizada.

Uma classificação comum em jogos considera com quantos jogadores uma pessoa pode interagir simultaneamente agrupando assim, os jogos em *singleplayer*, *multiplayer* e *massively multiplayer online*. Nos jogos *singleplayer* um jogador não interage com outros jogadores. Na segunda categoria estão os jogos *multiplayer*, que permitem uma experiência com vários jogadores, podendo ser em forma de disputa ou cooperação, e normalmente proporciona uma forma de comunicação social que está quase sempre ausente em jogos *singleplayer*. Os jogos *multiplayer* podem ser jogados tanto em rede quanto localmente.

Por último, também existem os *massively multiplayer online games* (MMOG) que são capazes de suportar grandes quantidades de jogadores simultaneamente em uma mesma partida. É importante ressaltar que essa classificação considera o número de jogadores interagindo ao mesmo tempo, portanto, mesmo que existam 7,5 milhões de pessoas

jogando simultaneamente o jogo *League of Legends*[5] (LoL) por existir um limite máximo de 10 jogadores em um partida, LoL é considerado um jogo *multiplayer* e não um MMOG. Neste trabalho, são chamados de jogos distribuídos todos os jogos *multiplayer* em rede e os *massively multiplayer online*.

Em jogos *multiplayer*, os clientes mantêm cópias dos objetos presentes no mundo do jogo. Quando um jogador interage com algum desses objetos, todos os outros clientes que são afetados por essa ação devem ter o seu estado interno do jogo atualizado. Uma solução encontrada para este problema é a utilização de áreas de interesse (*Area of Interest* - AOI). O jogador só recebe as informações que são relevantes sob seu ponto de vista, normalmente considerando a posição e o campo de visão do avatar dentro do mundo do jogo. Dessa forma, os jogadores apenas mantêm réplicas dos objetos que são interessantes para eles. Neste trabalho consideramos que a área de interesse é modelada como uma esfera que envolve o avatar do jogador. Desta forma, o jogador pode receber informações inclusive sobre objetos que não são visíveis.

Apesar da grande quantidade de gêneros e estilos dos jogos atuais, além dos avatares, existem apenas três tipos de entidades com as quais o jogador pode interagir: os objetos imutáveis, os objetos mutáveis e os personagens não jogáveis (NPCs - *Non Playable Character*) [Yahyavi and Kemme 2013]. Os objetos imutáveis são objetos criados *offline* e que nunca são alterados durante o jogo. Esses objetos normalmente são inicializados quando o jogo começa, por exemplo, objetos de cenário, o mapa do jogo, os modelos dos inimigos, dentre outros. Os objetos mutáveis são instâncias de objetos com os quais o jogador pode interagir e usar em si mesmo ou em outros jogadores. Exemplos desses objetos são poções que o jogador pode utilizar para recuperar a saúde de seu avatar. Os NPCs (personagens não jogáveis) são personagens ou ainda avatares que não são controlados por um jogador, são normalmente controlados por uma inteligência artificial. São muito utilizados para contar uma história, entregar missões ao jogador ou simplesmente trazer mais realismo para o ambiente fazendo ações rotineiras.

3. Técnicas de Predição de Movimento em Jogos

Um dos grandes problemas nos jogos *multiplayer* é garantir a consistência da visão dos multiplos jogadores. Por exemplo, considere dois avatares A e B. O avatar A está parado, enquanto o avatar B executa um movimento retilíneo uniforme de direção perpendicular à direção de A. Após alguns instantes, o avatar B muda sua direção de movimentação. O jogador B precisa enviar uma mensagem ao jogador A para informar sua nova posição.

Em jogos *Massively Multiplayer Online*, alguns com milhões de jogadores simultaneos, a quantia de mensagens transmitidas pode ser muito significativa. Técnicas de predição de movimento podem ser usadas para evitar o envio destas mensagens. Um exemplo de técnica preditiva é o *Dead Reckoning*.

O *Dead Reckoning* estima a nova posição tendo em vista a posição anterior e equações de movimento [Standards Committee on Interactive Simulation 1995]. Ambos os jogadores executam o algoritmo e uma mensagem só é enviada se o erro for superior ao um valor aceitável. As variáveis mais importantes para o *Dead Reckoning* são: a última posição conhecida x_t da entidade, a velocidade $v_t = \dot{x}_t$, a aceleração $a_t = \ddot{x}_t$ e t

[5]League of Legends: http://br.leagueoflegends.com/

que representa o instante de tempo. Outras informações que ajudam a estimar as forças que atuam nessa entidade também podem ser incluídas. Para objetos representados em um espaço tridimensional também são considerados a orientação, a velocidade angular e possivelmente a aceleração angular. A partir da definição dessas variáveis a trajetória de um objeto pode ser descrita utilizando as equações de movimento, mais especificamente pela da segunda lei de Newton, que utiliza a aceleração a_t de um objeto, a massa m e as forças F_i que esse objeto está submetido. Podendo assim fazer uma estimativa da posição de um determinado jogador.

Utilizando essas estimativas, cada jogador deve manter além de sua posição a posição estimada pelo *Dead Reckoning*. Quando a posição real e a posição estimada estiverem com uma diferença maior que o limiar aceitável, o jogador deve comunicar aos outros jogadores sua posição atual real. Cada jogador também deve manter um modelo de posição/orientação para todas as outras entidades que estiverem dentro de sua área de interesse. Essas posições/orientações estimadas das outras entidades são utilizadas para renderizar a posição/orientação das entidades no jogo. Porém, em jogos é muito comum que as regras físicas sejam relaxadas e mudanças drásticas de movimento são permitidas. É importante ressaltar que os clientes só recebem informações sobre os objetos que estão dentro de sua AOI em MMOGs.

Em [Yahyavi et al. 2012] é apresentado o algoritmo preditivo *AntReckoning* que é inspirado em colônia de formigas e tem como objetivo a utilização de modelos de interesse para prever a movimentação dos jogadores. As principais contribuições do *AntReckoning* são a incorporação do interesse do jogador na equação de movimento utilizada pelo *Dead Reckoning* e prover um arcabouço que utiliza feromônios para a modelagem considerando aspectos temporais e espaciais do interesse dos jogadores. Para incorporar o interesse dos jogadores na estimativa da posição desses avatares foi criado um modelo de forças de atração e repulsão. A intensidade das forças exercidas pelos pontos de atração (*Point of Interest* - POI) dependem de sua atratividade que pode ser determinada ou ainda aprendida. Esses pontos de interesse variam de um jogo para outro, mas um exemplo são os itens no jogo que atraem os jogadores, especialmente se os itens forem valiosos ou caso o jogador precise deles urgentemente. Da mesma forma, locais que não são seguros ao jogador, ou ainda locais em que existam inimigos muito fortes comparados ao nível do avatar.

Esses POIs são tratados pelo *AntReckoning* como formigas que geram feromônios para modelar a atração dos jogadores. Ao longo do tempo esses feromônios se propagam pelo mundo do jogo e a concentração acaba decaindo. Assim, naturalmente, fatores temporais e a geometria do mundo do jogo são simulados.

Assim como é comum em muitos jogos, o *AntReckoning* assume que o mundo do jogo é divido em células que não se sobrepõem. Denota-se por C o tamanho da célula na unidade do mundo do jogo. A gerência dos feromônios e as forças de atração exercidas por eles dependem da granularidade da célula: para cada avatar P, o cliente Q executa o algoritmo de *Dead Reckoning*, Q calcula a concentração de feromônio em cada célula e a respectiva soma das forças exercidas. Por questão de escalabilidade, apenas as células de uma região limitada em volta de P, chamada de zona de atração e denotada por R, é considerada no cálculo. É importante ressaltar que como os feromônios se propagam, mesmo POIs que não estão dentro de R são considerados. Outra questão é o fato de que

mesmo no *AntReckoning* o jogador também precisa executar o algoritmo para si mesmo, já que precisa saber quando deve se comunicar com os outros jogadores sobre seu posicionamento, considerando o limiar η escolhido.

4. Aprendizado de Máquina

Aprendizado de máquina é um ramo da inteligência artificial que faz intersecção entre a computação e a estatística, que busca compreender quais são as leis fundamentais que regem os processos de aprendizado com o objetivo de construir sistemas que aprendam para melhorar seus resultados de forma automática [Mitchell 2006].

Os algoritmos de aprendizado de máquina podem realizar tarefas tais como: Classificação, Predição, Clusterização e Associação [Camilo 2009]. As tarefas de Classificação e Predição são vistas como tarefas supervisionadas, enquanto as tarefas de Clusterização e Associação são tarefas não supervisionadas. As tarefas de Classificação têm como objetivo catalogar valores de variáveis qualitativas. Quando se busca estimar um valor numérico de uma variável a tarefa pode ser chamada de Regressão. As tarefas de Predição têm como objetivo prever um valor possível para uma variável no futuro. As predições numéricas tem como objetivo final a previsão para variáveis contínuas; no caso de variáveis discretas, devem ser utilizadas técnicas de classificação.

As tarefas de Agrupamento funcionam como descrito a seguir. Dado um conjunto de dados de entradas, são construídos agrupamentos, também chamados de *clusters*, que contém os registros mais semelhantes. Portanto, um dado é considerado similar aos elementos pertencentes ao mesmo *cluster* e, normalmente, para o cálculo da similaridade entre as entradas são utilizadas medidas de distâncias tradicionais. Por último, a Associação consiste em identificar o relacionamento dos ítens mais freqüentes em um determinado conjunto de dados.

4.1. Matriz de Confusão

A partir da obtenção dos resultados apresentados por um algoritmo de aprendizado de máquina, é necessário fazer um teste para verificar o quão confiável é o modelo. A matriz de confusão, apresentada em [Provost et al. 1997], contém informações sobre como as entradas foram classificadas, determinando se o valor previsto corresponde ao valor correto. Para obter uma matriz de confusão, é apresentado um conjunto de dados com suas classificações e para cada dado apresentado se obtém a classificação predita e compara-se com a classificação real. Todos os dados são contabilizados e os totais são apresentados em uma matriz. A matriz de confusão de um classificador binário é apresentado na tabela 1. Nessa matriz temos as colunas apresentando em quais categorias os valores foram classificados pelo algoritmo, enquanto nas linhas temos as categorias que essas instâncias realmente pertencem. É possível observar que existem duas categorias em que as instâncias podem estar e existem quatro grupos em que uma instância pode ser representada nessa matriz. As instâncias que pertencem aos grupos *a* e *d* foram corretamente classificadas. Já as instâncias pertencentes aos grupos *b* e c estão sendo classificadas de forma errônea.

As instâncias pertencentes aos grupos de *a*, *b*, *c* e *d* são chamadas respectivamente de Verdadeiras Negativas (True Negatives - TN); Falsas Positivas (False Positives - FP); Falsas Negativas (False Negatives - FN); e Verdadeiras Positivas (True Positives - TP).

	Classificado: Negativo	Classificado: Positivo
Realidade: Negativo	TN = a	FP = b
Realidade: Positivo	FN = c	TP = d

Tabela 1. Matriz de confusão exemplo.

5. A Abordagem Proposta

Nessa Seção é apresentada a proposta para a predição de movimento baseada em aprendizado de máquina. Inicialmente é descrito o jogo *World of Warcraft*. Em seguida, os *traces* reais utilizados são descritos bem como a nova base que foi construída baseada estes *traces*. Por fim, a nova estratégia é apresentada.

5.1. O Jogo *World of Warcraft* e a Cidade *IronForge*

Para realizar esse trabalho foi escolhido o jogo *World of Warcraft*[6] (WoW). Este jogo pertence ao gênero de MMOG e constitui uma grande parcela do mercado deste gênero com 100 milhões de contas criadas, sendo jogado em 224 países e territórios, segundo a *Blizzard Entertainment* [7]. Esse gênero tem algumas características intrínsecas: a necessidade de uma maneira de medir quantitativamente o progresso do jogador, a interação social com outros jogadores, a possibilidade de personalização dos avatares e uma arquitetura do sistema construída focando a presença de um grande volume de jogadores. Em WoW, os jogadores podem observar o seu progresso no jogo por meio do avanço do nível do avatar, nos pontos gastos em habilidades, em equipamentos adquiridos, na riqueza obtida dentro do jogo ou ainda pelas de conquistas obtidas fazendo certas tarefas. Para progredir nesse jogo, os avatares batalham com monstros ou ainda fazem missões que são dadas por NPCs. Ambas ações podem ser feitas em grupo ou de forma solitária e normalmente provêm recompensas como pontos de experiência que são utilizados para que os avatares subam de nível.

Em particular a predição de movimento é feita utilizando dados colhidos na cidade de *Ironforge* devido a disponibilidade de traces do jogo nesta cidade. A cidade de *Ironforge* é considerada uma cidade capital da Aliança no jogo *World of Warcraft*. Os jogadores de *World of Warcraft* podem participar de uma das duas facções presentes, a Aliança ou a Horda. Dependendo do servidor em que esses jogadores estejam jogando, é possível fazer batalhas do tipo jogador-versus-jogador entre jogadores de facções rivais. É importante ressaltar que como a cidade analisada é uma das capitais da Aliança, é improvável a presença de jogadores com avatares da Horda, e portanto geralmente o comportamento dos jogadores não assume formas agressivas. O que garante isso é que inclusive os NPCs da cidade irão atacar avatares da Horda caso sejam vistos, se tornando assim um ambiente extremamente agressivo aos avatares da Horda. Outra informação importante sobre *Ironforge* é que ela é a cidade inicial para todos os jogadores da raça anã, portanto, é uma cidade que é visitada tanto por avatares poderosos quanto por avatares fracos. Em *World of Warcraft*, a partir do nível 20, os jogadores podem utilizar montarias com uma velocidade maior do que o avatar caminha e acima do nível 60 os jogadores podem inclusive utilizar montarias que voam.

[6]http://us.battle.net/wow/pt/

[7]Blizzard Entertainment: http://us.battle.net/wow/pt/blog/12346804/world-of-warcraft-azeroth-by-the-numbers-1-28-2014

Os NPCs desta cidade podem desempenhar uma das seguintes funções: *Auctioneer, Banker, Battlemaster, Collector, Innkeeper, Merchant, Quartermaster, Quest Giver, Stable Master, Trainer, Weapon Master*. O interesse por qualquer um desses NPCs é muito individual de um jogador, pois um jogador que tenha a profissão de cozinheiro não se interessará pelo mercante que vende ítens para forja ou forneça treinamento em alfaiataria, por exemplo. Além disso, um jogador com um nível muito alto, não se interessará por missões para iniciantes.

5.2. *Traces* Reais do Jogo Utilizadas

Neste trabalho foram utilizados traces reais do jogo WoW disponíveis em duas bases publicas na Internet: *WoWPosition* e *WoWHead*. A partir dos dados destas bases, foi construída uma nova base utilizada no desenvolvimento da proposta apresentada no presente trabalho.

A base de dados *WoWPosition* [Shen et al. 2014] é uma das bases que foi utilizadas neste trabalho. Essa base de dados apresenta informações sobre a posição dos jogadores. Entretanto, apresenta unicamente esta informação, não tendo qualquer outro tipo de informação sobre nível, raça ou ainda classe de um determinado avatar. Destaca-se que essa base de dados não contém qualquer informação que também possa ser utilizada para identificar o avatar. Dentre os mapas que foram analisados para a construção desta base foi escolhida os dados do arquivo "Ironforge-1d". Essa base contém informações de 1302 jogadores, os quais foram coletadas na cidade de *Ironforge*.

A base de dados *WoWPosition* contém mais de 30.000 entradas e é composta por uma tabela com os seguintes campos: *timestamp, id, x, y, z e ângulo de visão*, descritos a seguir. O campo *timestamp* é a quantidade de segundos que se passaram desde do primeiro segundo do primeiro dia do ano de 1970. O campo *id* apresenta um número de identificação utilizado internamente para identificar um jogador. Os jogadores foram anonimizados e, portanto, não é possível extrair mais informações, além das fornecidas na própria base sobre os mesmos. Os campos *x, y e z* são os dados da posição do jogador dentro do jogo e o *ângulo de visão* representa para qual direção o jogador está olhando.

Outra base de dados que foi utilizada neste trabalho é a base *WowHead* [8] . A partir dessa base de dados é possível extrair informações sobre os NPCs presentes no mapa da cidade de *Ironforge*. Essa base de dados é composta por uma tabela com 4 campos: *Nome, ThottbotXPosition, ThottbotYPosition e FunçãoNPC*. O primeiro campo é composto pelo nome do NPC, o segundo e o terceiro campos são responsáveis pelo armazenamento da posição do NPC no mapa dentro do sistema de coordenadas do denominado *Thottbot* e o último campo descreve a função deste NPC. Foram criadas as seguintes categorias para descrever a função de um NPC dentro do jogo: *Auctioneer, Banker, Battlemaster, Collector, Innkeeper, Merchant, Quartermaster, Quest Giver, Stable Master, Trainer, Weapon Master*.

O jogo *World of Warcraft* é um jogo em 3 dimensões, entretanto a base *WoWHead* faz uma conversão da posição tridimensional de todos o NPCs para um mapa com 2 dimensões [9]. Por conta dessa característica a coordenada que representa a altura é ignorada.

[8]WoWHead: http://www.wowhead.com

[9]Coordenadas Thottbot: http://www.wowwiki.com/Mapcoordinates

O sistema de coordenadas utilizado consiste de uma tupla (x, y) com os valores variando de 0 até 100 e representam uma localização dentro deste mapa com duas dimensões. Nesse sistema, o canto superior esquerdo representa a posição $(0, 0)$ e o canto inferior direito representa a posição $(100, 100)$, dessa maneira, o centro do mapa é representado pela posição $(50, 50)$.

5.3. Construção da Nova Base

Utilizando as base de dados *WoWPosition* e *WoWHead* foi construída uma nova base que contém os seguintes campos: *ThottbotX*, *ThottbotY*, o *AnguloVisual*, *AnguloMovimentacaoAnterior*, os dados dos outros jogadores e NPCs presentes no campo de visão, o *AnguloMovimentacao* e *DeadReckoning*. Os campos *ThottbotX* e *ThottbotY* indicam a posição do jogador no sistema de coordenadas do *Thottbot*, enquanto o campo *AnguloVisual* é responsável por indicar para qual direção que o jogador que está sendo análisado está observando. Os campos com as informações dos outros jogadores e dos NPCs e serão descritos no próximo parágrafo. Os dois últimos campos são o *AnguloMovimentacao* e o *DeadReckoning*. O campo *AnguloMovimentacao* descreve qual o ângulo em que o jogador se movimentou no respectivo *timestamp* e o campo *DeadReckoning* indica se o algoritmo *Dead Reckoning* acerta aquele caso ou não. Em outras palavras, o campo *DeadReckoning* deve ser utilizado para o treinamento dos modelos de aprendizagem na primeira fase da nova estratégia. Enquanto o campo *ÂnguloMovimentacao* é utilizado durante a segunda fase da estratégia proposta. Portanto, quando se está na etapa de o aprendizado para determinar se o *Dead Reckoning* irá acertar ou não, durante a primeira fase, o campo *AnguloMovimentacao* é ignorado. Já quando está sendo feito o aprendizado do modelo da segunda fase, quando de fato é determinado o novo ângulo de movimentação do jogador, o campo *DeadReckoning* deverá ser ignorado.

Os campos dos jogadores e NPCs são: *X*, *Y*, *Distância*, *Angulo* e *Função*. Os campos *X* e *Y* apresentam a posição destes jogadores ou NPCs no sistema de coordenadas do *Thottbot*. O campo *Distância* mostra a distância entre o jogador analisado e um outro jogador ou NPC, enquanto o campo *Angulo* indica o ângulo entre os dois. O campo *Função* é exclusivo dos NPCs e mostra qual a função de um determinado NPC no jogo. A quantidade de campos presentes na base de dados precisa ser fixa, assim no máximo N jogadores e NPCs são tratados dentro do campo de visão de cada jogador. Caso existam mais que N jogadores ou NPCs no campo de visão, apenas os N mais próximos são tratados.

Para determinar se um avatar ou NPC está dentro do campo de visão de um jogador é necessário verificar o ângulo entre eles e qual é o ângulo que o jogador está observando. Os dados fornecidos pela base *WoWPosition* estão no sistema de posicionamento real utilizado pelo jogo, entretanto todos os NPCs têm o posicionamento apresentado pela base *WoWHead*. Portanto, para poder ser feita a análise do campo de visão de um jogador, todos os avatares tiveram suas coordenadas de posição convertidas para o sistema da base *WoWHead*. Destaca-se que esta decisão foi tomada considerando que na base *WoWHead* não existe a coordenada de altura. Outra informação relevante é que não temos qualquer informação sobre o posicionamento da câmera de cada jogador ou ainda sua direção de observação. O que temos é a informação sobre qual o ângulo de determinado avatar está rotacionado. Assim verificamos se um NPC ou avatar está dentro do campo de visão utilizando dois testes. Primeiramente, é testado se o avatar/NPC está dentro da AOI do

jogador. Em [Shen et al. 2014] é apresentado que a área de interesse dos jogadores é representada por um círculo de raio de 100 unidades do jogo, essa medida está sendo utilizada para o valor de d. Em seguida, é testado se o avatar/NPC foi utilizado o valor α para determinar o ângulo de abertura do campo de visão do jogador. Neste trabalho foi utilizado o valor de 80º para α por ser próximo ao valor do ângulo de abertura do campo de visão de seres humanos.

Foi realizada uma "limpeza da base", utiliando os modelos de aprendizado construídos visando eliminar campos ruidosos e aqueles que não podem ser interpretados individualmente. Portanto, a base de dados ficou com os seguintes campos: *AnguloVisual*, *AnguloMovimentaçãoAnterior*, *AnguloMovimentacao* e *DeadReckoning* e dados dos jogadores e dos NPCS são compostos por *Distancia*, *Angulo* e *Funcao*.

5.4. Estratégia Proposta

A estratégia proposta deve ser utilizada sempre que for necessário renderizar um determinado avatar de um jogador por outro. A estratégia proposta consiste em duas fases bem definidas. Na primeira fase, algoritmos de aprendizado de máquina são utilizados para gerar um modelo que classifica se o *Dead Reckoning* acerta ou erra na predição de movimento. O *Dead Reckoning*, considera que um jogador se movimenta em linha reta e utiliza regras físicas para prever a nova posição. Caso contrário, isto é, quando a primeira fase indica mudança de direção é executada a segunda fase. Na segunda fase, algoritmos de aprendizado de máquina são utilizados para gerar um modelo que faz a predição do ângulo de movimento do avatar.

Para implementar a estratégia foi utilizada a ferramenta Weka [10] (Waikato Environment for Knowledge Analysis) em sua versão 3.6.12. Foram escolhidos experimentalmente quatro algoritmos, apresentados a seguir, para fazer a predição de movimento: *Reduced Error Pruning Tree* (REPTree), *Local Weighted Learning* (LWL), Bootstrap Aggregating (*Bagging*) e uma rede *Multilayer Perceptron*.

O algoritmo *Reduced Error Pruning Tree* (REPTRee) é baseado em árvores de decisão [Quinlan 1987]. O algoritmo *Local Weighted Learning* (LWL) faz um sistema de votação ponderada [Atkeson et al. 1997]. O *Bootstrap Aggregating* (*Bagging*) usa a estratégia de gerar várias versões de um mesmo classificador utilizando diferentes versões do conjunto de treinamento original [Opitz and Maclin 1999]. Por fim, foi utilizada uma rede neural do tipo *Multilayer Perceptron*. [Ripley 1996].

6. Resultados Experimentais

Inicialmente foram criadas duas bases de dados com os dados de um período de tempo de 1 hora cada: A primeira base trata da primeira hora e a segunda base tratou da oitava hora. Foi utilizado o valor N igual a 7. As informações dessas bases foram sintetizadas na tabela 2. Essa tabela é composta por sete campos. O primeiro campo identifica a base e o segundo campo indica o total de entradas presente nesta base. O segundo e o terceiro campo indicam respectivamente o total de acertos e erros do algoritmo *Dead Reckoning* nesta base. O quarto campo indica quantos avatares distintos estão sendo retratados nesta base de dados. Os últimos dois campos indicam o *Timestamp* inicial e final que foram utilizados para a construção destas bases de dados.

[10] http://www.cs.waikato.ac.nz/ml/weka/

Identificação da base	Total de Entradas	Total de acertos	Total de erros	Avatares distintos	Timestamp Inicial	Timestamp Final
1	19870	61,17%	38,83%	164	63445137060	63445140659
2	4885	44,54%	2709 (55,46%	43	63445165860	63445169459
Somatório	24755	57,89%	10425(42,11%	200	-	-

Tabela 2. Informações sobre as bases de dados geradas.

Através da tabela 2 é possível retirar algumas informações sobre o comportamento do *Dead Reckoning*. Nessas bases a taxa de acerto médio do algoritmo *Dead Reckoning* é de 57,89%, e , portanto, a taxa de erro é de 42,11%.É importante ressaltar que cada uma das entradas presentes na base de dados equivale ao registro da posição de um avatar dentro do jogo *World of Warcraft*. Como apresentado anteriormente, as situações em que o *Dead Reckoning* erra sua previsão, são as situações em que devem haver troca de mensagens comunicando a nova direção. Portanto, nessas bases de dados em 42.11% entre um segundo e outro houve ao menos uma troca de mensagens para comunicar a mudança da movimentação de jogadores.

Além do comportamento do algoritmo *Dead Reckoning*, é possível também observar que os jogadores têm um comportamento diferente dependendo do horário. Observa-se que na tabela 2 é significativa a diferença entre a quantidade de avatares distintos entre a primeira e a oitava hora apesar de ter sido utilizada a mesma quantidade de timestamps. Na primeira hora, que foi utilizada para construir a base de dados 1, existem 164 avatares distintos. Enquanto na oitava hora, que foi utilizada para a construção da base 2, existem apenas 43 avatares distintos. Como havia menos jogadores é esperado que existam menos dados de movimentação na base 2 do que na 1.

São apresentados resultados de três experimentos. O primeiro experimento têm por objetivo determinar quando *Dead Reckoning* erra ou acerta. O segundo experimento tinha como objetivo determinar, nos casos em que o *Dead Reckoning* erra, qual será o novo ângulo de movimentação. No terceiro experimento, a segunda fase foi executada apenas utilizando os dados que foram classificados como Verdadeiros Positivos e Falsos Negativos na primeira fase.

6.1. Primeiro Experimento

O primeiro experimento tinha como objetivo verificar se é possível utilizar algoritmos de aprendizado de máquina para classificar quando o algoritmo *Dead Reckoning* erra a previsão. Os resultados do primeiro experimento são apresentados na tabela 3. É possível observar que, em média os algoritmos obtiveram uma taxa de acerto de 79,29%. Além disso, é possível observar que o algoritmo *Bagging* obteve o melhor resultado ao obter uma taxa de acerto de 83,17%. Ressalta-se que os algoritmos, em média, classificam 26,59% das entradas em que o *Dead Reckoning* acertaria de forma errada, ou seja, 26,59% das entradas são Falsos Positivos. Dessa forma, 26,59% dos casos em que o *Dead Reckoning* acertaria na predição são enviados para a segunda fase da estratégia proposta de forma incorreta. Também observa-se que em 20,70% das vezes em que o *Dead Reckoning* erra a predição, os algoritmos de aprendizado apontam que *Dead Reckoning* irá acertar. Assim, para todas essas entradas a predição será errada tanto utilizando o *Dead Reckoning* quanto na nova estratégia. A conclusão obtida neste experimento é que com a média da taxa de acerto entre os algoritmos sendo 76.60% verificou-se que é viável utilizar aprendizado de máquina para a primeira fase do modelo.

Algoritmo	Treinamento	Teste	TP	FP	TN	FN	Taxa de acerto
MultilayerPerceptron	1	2	81,35%	22,98%	77,02%	18,65%	79,42%
Bagging com REPTree	1	2	83,17%	21,43%	78,57%	16,83%	81,10%
LWL	1	2	69,92%	39,82%	61,18%	30,08%	65,85%
REPTree	1	2	82,74%	23,16%	76,84%	17,26%	80,04%
Média	-	-	79,29%	26,59%	73,40%	20,70%	76,60%

Tabela 3. Resultados interpretados da aprendizagem do campo DeadReckoning.

6.2. Segundo Experimento

O segundo experimento tinha como objetivo determinar se é possível utilizar algoritmos de aprendizado de máquina para prever qual será o novo ângulo de movimentação de um jogador, quando o jogador muda de direção. Para isso foram removidas todas as entradas em que o *Dead Reckoning* faz a previsão correta. Neste experimento foi considerado que uma saída está correta se houver um erro menor que 15º, e portanto menor que 5%. Os resultados obtidos pelos algoritmos para a predição do novo ângulo de movimento podem ser observados na tabela 4. Nessa tabela também é possível observar que os algoritmos erraram 43,48% de todas as entradas que deveriam ir para segunda fase. Em outras palavras, os algoritmos de aprendizado erraram na predição de qual será o novo ângulo de movimentação em 43,48% dos casos. Como os algoritmos estão sendo utilizados para fazer uma predição, é possível observar que a média do erro absoluto foi de 29,62%. Novamente, como foi no teste anterior, o algoritmo de *Bagging* obteve o melhor desempenho, tendo um erro médio absoluto de 15,35% e ele conseguiu prever corretamente para 83,78% das entradas o novo ângulo de movimento. Estes resultados demonstram a viabilidade do aprendizado do novo ângulo xdos jogadores no jogo *World of Warcraft*.

Algoritmo	Treinamento	Teste	Erro Absoluto	Coef. de Cor.	Saídas Corretas	Saídas Erradas
MultilayerPerceptron	1	2	33,44 %	0,81	1067(49,04%)	1109(50,96%)
Bagging com REPTree	1	2	15,35 %	0,92	1823(83,78%)	353(16,22%)
LWL	1	2	52,50 %	0,79	230 (10,56%)	1946(89,44%)
REPTRee	1	2	17,20 %	0,91	1799(82,67%)	377(17,33%)
Média	-	-	29,62 %	0,85	56,50%	43,48%

Tabela 4. Resultados para aprendizagem do campo *AnguloMovimento*.

6.3. Terceiro Experimento

O terceiro experimento foi elaborado considerando que se o classificador errar durante a primeira fase, mas a previsão na segunda fase estiver correta, a saída estará correta. Assim foram construídas bases de dados utilizando todas as entradas em que cada um dos algoritmos classificou como entradas que deveriam ir para a segunda fase. Ou seja, para cada um dos algoritmos foi construída uma base utilizando as entradas Falsas Negativas quanto Verdadeiras Positivas indicadas pelos próprios modelos da primeira fase.

Observa-se que nesta tabela que os algoritmos erram em média em 48,97% de todas as entradas que foram para segunda fase. No teste anterior, foi observado que eles erravam em 43,48% dos casos. Portanto, houve uma variação de 5,49% na quantidade de saídas erroneamente classificadas. Observa-se também que o erro absoluto médio obtido durante o terceiro teste é de 34,88%, enquanto no segundo teste foi de 29,62%. Assim a variação foi de 5,26% entre o segundo e o terceiro teste.

Algoritmo	Total de Entradas	Erro Absoluto	Coef. de Cor.	Saídas Corretas	Saídas Erradas
MultilayerPerceptron	2167	36,75 %	0,76	1021 (47,12%)	1146 (52,88%)
Bagging com REPTree	2193	21,86 %	0,87	1609 (73,37%)	584 (26,63%)
LWL	2272	55,79 %	0,67	312 (13,73%)	1960 (86,26%)
REPTRee	2237	25,12 %	0,85	1563 (69,87%)	674 (30,13%)
Média	2717.25	34,88 %	0,78	51,02%	48,97%

Tabela 5. Resultados obtidos no terceiro teste.

Algoritmo	1a Fase		2a Fase	
	Caso 1	Caso 2	Caso 3	Caso 4
MultilayerPerceptron	2211 (45,26%)	507 (10,37%)	1021 (20,90%)	1146 (23,45%)
Bagging com REPTree	2239 (45,83%)	453 (9,27%)	1609 (32,93%)	584 (11,95%)
LWL	1827 (37,40%)	786 (16,09%)	312 (6,38%)	1960 (40,12%)
REPTRee	2191 (44,85%)	457 (9,35%)	1563 (31,95%)	674 (13,79%)
Média	43,33%	11,27%	23,04%	22,32%

Tabela 6. Resultados obtidos para as entradas presentes na base 2.

Na tabela 6 são apresentados os resultados de como uma instância pode ser classificada utilizando esta estratégia. O casos contabilizados como Caso 1 são casos que o modelo indica que o *Dead Reckoning* irá acertar e de fato ele acerta. Já os casos classificados como Caso 2 são casos que o modelo indica que o *Dead Reckoning* irá acertar, porém ele erra. Durante a segunda fase, o modelo pode prever corretamente a nova direção do jogador ou não, estes casos estão sendo computados respectivamente pelo Caso 3 e Caso 4 na tabela. Desta forma, é possível observar na tabela 6, que o algoritmo *Bagging* com o algoritmo REPTree obteve o melhor resultado se comparado a todos os outros algoritmos. De todas as entradas presentes na base 2, o algoritmo *Bagging* previu corretamente, na primeira fase, que 45.83% das entradas o algoritmo *Dead Reckoning* seria suficiente. Além disso, de todas as entradas na base 2, o algoritmo *Bagging* previu corretamente o novo ângulo de movimentação para 32,93% de todas as entradas fornecidas. Desta maneira, somando a porcentagem de casos 1 e 3 temos uma taxa de acerto de 78,76% de todas as entradas presentes na base 2. Destaca-se que nesta base, como foi apresentado na tabela 2, o algoritmo *Dead Reckoning* obteve uma taxa de acerto de 44,54%.

7. Conclusão

Neste trabalho foi apresentada uma proposta de predição da movimentação de avatares em jogos distribuídos *multiplayer* baseada em aprendizado de máquina. Para a validação da estratégia proposta foi construída uma base de dados, baseada em traces do jogo *World of Warcraft*, a partir das bases disponíveis publicamente *WoWPosition* e *WoWHead*. A nova base de dados construída contém informações sobre jogadores e NPCs que estão presentes no campo de visão de um determinado jogador. Além destas informações, também estão disponíveis dados sobre o posicionamento deste jogador.

Observou-se, que a estratégia proposta obtém uma taxa de acerto médio de 76,60% durante a primeira fase. Na segunda fase obtém uma taxa de acerto de 51,02%. O algoritmo de aprendizagem de máquina que obtém o melhor desempenho é o Bagging. Esse algoritmo obtém uma taxa de acerto durante a primeira e a segunda fase respectivamente de 81,10% e 73,37%. Estes resultados mostram que a utilização de aprendizado de máquina para a predição da movimentação de avatares jogos distribuídos é uma alternativa muito atraente Em comparação com uma outra abordagem relacionada da literatura, o AntReckoning, a taxa de sucesso é semelhante, cerca de 30% de melhoria sobre o *Dead*

Reckoning. Entretanto, a grande vantagem da estratégia proposta é que não necessita de um especialista para configurar os múltiplos parâmetros utilizados por aquele modelo. Utilizando aprendizado de máquina, os parâmetros são ”automaticamente aprendidos”.

Entre os trabalhos futuros um item importante é construir uma base de dados ainda com mais informações que a construída, melhorando a predição. Além disso, os modelos propostos devem ser aplicados para outros jogos distribuídos.

Referências

Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally weighted learning for control. In *Lazy learning*, pages 75–113. Springer.

Camilo, C. O. (2009). Mineração de dados: Conceitos, tarefas, métodos e ferramentas.

Claypool, M. and Claypool, K. (2010). Latency Can Kill: Precision and Deadline in Online games. In *Proceeding MMSys '10 Proceedings of the first annual ACM SIGMM conference on Multimedia systems*.

Mitchell, T. M. (2006). *The discipline of machine learning*, volume 17. Carnegie Mellon University, School of Computer Science, Machine Learning Department.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, pages 169–198.

Provost, F., Fawcett, T., and Kohavi, R. (1997). The case against accuracy estimation for comparing induction algorithms. In *In Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann.

Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234.

Ripley, B. D. (1996). *Pattern recognition and neural networks*. Cambridge university press.

Shen, S., Brouwers, N., Iosup, A., and Epema, D. (2014). Characterization of human mobility in networked virtual environments. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, page 13. ACM.

Standards Committee on Interactive Simulation (1995). IEEE Standard for Distributed Interactive Simulation - Application Protocols . In *Proceedings of IEEE Standard 1278.1-1995*, New York, NY, USA.

Yahyavi, A., Huguenin, K., and Kemme, B. (2012). Interest modeling in games: the case of dead reckoning. *Multimedia Systems*, pages 255–270.

Yahyavi, A. and Kemme, B. (2013). Peer-to-Peer Architectures for Massively Multiplayer Online Games: A Survey. *ACM Computing Surveys*, 46 Issue 1:Article 9.

Trilha Principal do SBRC 2016
Sessão Técnica 13
Gerenciamento e Roteamento em SDN

Roteamento Multicaminhos em Redes Definidas por Software

Pedro H.A. Rezende[1], Luís F. Faina[1], Lásaro Camargos[1] e Rafael Pasquini[1]

[1]Faculdade de Computação (FACOM) – Universidade Federal de Uberlândia (UFU)
Caixa Postal 593 – 38.408-100 – Uberlândia – MG – Brazil

pedroh@mestrado.ufu.br,{luisfaina,lasaro,rafael.pasquini}@ufu.br

Resumo. *Este artigo apresenta e avalia um mecanismo para suportar roteamento multicaminhos em Redes Definidas por Software (SDN) OpenFlow. Utilizando informações do monitoramento de vazão e atraso para fluxos, tendo por base consultas explícitas a contadores OpenFlow mantidos nos elementos de rede, o modulo proposto de multicaminhos é capaz de ramificar fluxos na rede, de tal forma a atender demandas de largura de banda. O mecanismo multicaminhos proposto, diferentemente do MPTCP (MultiPath TCP), não requer nenhuma atualização na pilha de protocolos existente nos sistemas finais. Por ser uma solução desenvolvida diretamente no controlador OpenFlow, é capaz de tratar qualquer fluxo utilizando a rede. Os resultados apresentados neste artigo demonstram o funcionamento do mecanismo de multicaminhos, evidenciando o comportamento da solução frente a existência ou não de tráfego concorrente através dos enlaces compondo a rede.*

Abstract. *This paper presents and evaluates a mechanism to provide multipath routing in OpenFlow Software Defined Networks. Based on monitoring information regarding throughput and flow latency, gathered from a polling mechanism which reads counters maintained at OpenFlow network elements, the proposed multipath module is capable of splitting flows across disjoint paths in the network, in order to satisfy bandwidth demands. The proposed multipath module, differently from MultiPath TCP, does not require changes in the protocol stack of end-hosts. Once such solution is completely developed in an OpenFlow controller, it is able to handle any flow using the network. This paper details the multipath module workings, highlighting its behavior in the presence or absence of concurrent traffic in the links composing the network.*

1. Introdução

Embora haja abundância de caminhos na Internet, um fluxo é, em geral, encaminhado por um único caminho. Alternativamente, o roteamento multicaminhos permite que o tráfego seja encaminhado redundantemente ou com maior vazão fim-a-fim, proporcionando melhor uso dos recursos da rede, segurança, bem como melhor qualidade de serviço.

Equal-Cost-MultiPath (ECMP) [Thaler 2000] e *MultiPath* TCP (MPTCP) [Ford et al. 2013] são soluções bem conhecidas para prover multicaminhos. A primeira determina a interface de egresso de um fluxo por meio do *hash* do cabeçalho dos pacotes compondo-o. Assim, mesmo que diferentes fluxos entre as mesmas entidades fim-a-fim sigam caminhos distintos, todo o tráfego para um dado fluxo tende a seguir o mesmo caminho. O MPTCP, por sua vez, divide um fluxo TCP em sub-fluxos, o que permite

que cada sub-fluxo seja, do ponto de vista lógico, transmitido por um caminho distinto. Contudo, o MPTCP é centrado nos sistemas finais e mesmo alterando a pilha de protocolo destes elementos, não há garantia alguma de que estes sub-fluxos serão, de fato, encaminhados através de multicaminhos disjuntos do ponto de vista físico.

Recentemente, Redes Definidas por Software (*Software Defined Networks* - SDN) vêm chamando a atenção da indústria e academia. SDN separa os planos de controle, responsável pela tomada de decisões de roteamento de fluxos, e de dados, responsável pelo encaminhamento dos fluxos de acordo com as decisões do plano de controle.

OpenFlow [Foundation 2013] é, provavelmente, a arquitetura SDN mais amplamente conhecida e pesquisada. Além da separação entre os planos de controle e dados, o OpenFlow contempla contadores estatísticos de diferentes métricas (e.g., números de bytes e pacotes recebidos/transmitidos) e granularidades (e.g., por fluxo, por grupo ou por interface), além das mensagens para a coleta destas métricas nos elementos de rede.

Em trabalho anterior [Rezende et al. 2015], fundamental para o mecanismo de multicaminhos proposto neste artigo, desenvolveu-se um módulo de monitoramento para um controlador OpenFlow. O módulo de monitoramento enriquece, de maneira precisa, a visão do controlador referente ao estado atual da malha de elementos de rede, incluindo aferições referentes a atraso e consumo de largura de banda em diversas granularidades.

Uma vez que o plano de controle OpenFlow, materializado na figura de um controlador que contempla o módulo de monitoramento apresentado em [Rezende et al. 2015, Rezende 2016], tem uma visão global e enriquecida da rede, o mesmo pode ser utilizado na obtenção de um mecanismo de roteamento multicaminhos que leve à melhor utilização de recursos da rede, de forma transparente às aplicações. Este é o escopo de investigação deste artigo.

Este artigo propõe um mecanismo para roteamento multicaminhos em OpenFlow, bem como avalia os resultados alcançados através de um protótipo. Este artigo também apresenta como o mecanismo proposto para o roteamento multicaminhos está integrado ao módulo de monitoramento apresentado e validado no trabalho anterior [Rezende et al. 2015, Rezende 2016].

Experimentos exploram o mecanismo proposto de multicaminhos, evidenciando seu comportamento na presença e na ausência de tráfego concorrente. Os resultados mostram o ganho de tempo ou de largura de banda quando da adoção de multicaminhos para o encaminhamento de fluxos. O protótipo foi avaliado durante a transferência de arquivos com o SCP e através da geração de tráfego sintético a partir do gerador IPerf.

O restante deste artigo está organizado com se segue. Seção 2 apresenta os trabalhos relacionados. Seção 3 detalha o mecanismo de multicaminhos proposto. Seção 4 discute os experimentos, resultados e análises dos dados obtidos do protótipo desenvolvido. Finalmente, a Seção 5 resume os conclusões do trabalho.

2. Trabalhos Relacionados

[Sandri et al. 2015] propõe o uso do MPTCP em conjunto com a arquitetura SDN. Na abordagem proposta pelos autores, um controlador SDN é utilizado para garantir que os sub-fluxos gerados pelo MPTCP sempre passem por caminhos disjuntos. Entretanto, de acordo com a proposta dos autores, o *host* precisa definir à *priori* quantos sub-fluxos

serão usados pelo MPTCP. Caso o *host* requisite um novo caminho e a rede não possa disponibilizar mais caminhos disjuntos, o controlador irá recusar essa requisição.

Em [van der Pol et al. 2012] os autores apresentam um mecanismo de multicaminhos baseado no MPTCP e OpenFlow. Os autores usam o OpenFlow para descobrir a topologia da rede e aplicar regras de encaminhamento baseadas em um conjunto fixo de 8 identificadores de VLANs. O MPTCP está configurado para particionar o tráfego total em 8 sub-fluxos e o OpenFlow atribui um caminho disjunto para cada um dos 8 sub-fluxos.

A nossa proposta, apresentada neste artigo, difere-se dos trabalhos apresentado em [Sandri et al. 2015, van der Pol et al. 2012] por ser independente de protocolo, suportando o tratamento de multicaminhos para todos os fluxos que utilizam a rede OpenFlow, sem que haja a necessidade de alteração na pilha de protocolos dos *hosts*. Além disso, a definição dos caminhos a serem utilizados é automaticamente obtida pelo módulo multicaminhos proposto neste artigo, através de informações precisas sobre o plano de dados, obtidas pelo módulo de monitoramento que apresentamos anteriormente em [Rezende et al. 2015].

[Jinyao et al. 2015] propõe um *framework*, chamado HiQoS, para prover encaminhamento de dados sobre múltiplos caminhos usando a arquitetura SDN. O *framework* possui dois componentes, o de serviços diferenciados e o de roteamento em múltiplos caminhos. O primeiro componente é responsável por classificar o tráfego das aplicações de acordo com o uso estimado de largura de banda. As classificações possíveis são: vídeo com alto uso de banda; vídeo/áudio com baixo atraso; e fluxo de dados como serviço de melhor esforço. O módulo de roteamento consulta os dispositivos periodicamente, coletando a largura de banda utilizada em cada fila das interfaces dos elementos de rede. A partir dessa métrica, verifica qual a fila menos utilizada para aplicar o novo fluxo.

Em [Li and Pan 2013] é proposto um balanceador de carga para redes de data centers que suportem o protocolo OpenFlow. O balanceador de carga, proposto pelos autores como uma extensão de um controlador OpenFlow, consulta os switches periodicamente para coletar a largura de banda de cada enlace. Através da coleta da largura de banda, o balanceador define o encaminhamento para um fluxo, sendo que a construção do caminho ocorre de maneira recursiva. Em cada chamada recursiva é escolhido um switch, através da descoberta do enlace com maior largura de banda disponível e da restrição do nível de acesso. Após a definição de um caminho, o controlador aplica as regras nos switches e o fluxo passará por este caminho até o seu término.

Diferentemente da proposta apresentada neste artigo, as escolhas dos melhores caminhos em [Jinyao et al. 2015, Li and Pan 2013] baseiam-se na divisão em nível de fluxo, ou seja, um fluxo é encaminhado pelo mesmo caminho até o seu término, não havendo a divisão de um mesmo fluxo em múltiplos sub-fluxos.

3. Arquitetura de Roteamento Multicaminhos

A Arquitetura SDN contempla três planos: aplicação, controle, e dados. Embora esta estrutura seja amplamente conhecida e contemple a API *Northbound* entre os planos de aplicação e controle, e a API *Southbound* entre os planos de controle e dados, apenas esta última API possui uma implementação bem estabelecida, realizada no OpenFlow.

A Figura 1 detalha a Arquitetura SDN. Os módulos em branco compõem a estru-

tura original do OpenFlow, enquanto os módulos hachurados compõem as adições propostas aqui para habilitar a solução de roteamento multicaminhos em SDN. Como pode ser observado, a adição de componentes se dá nos três planos.

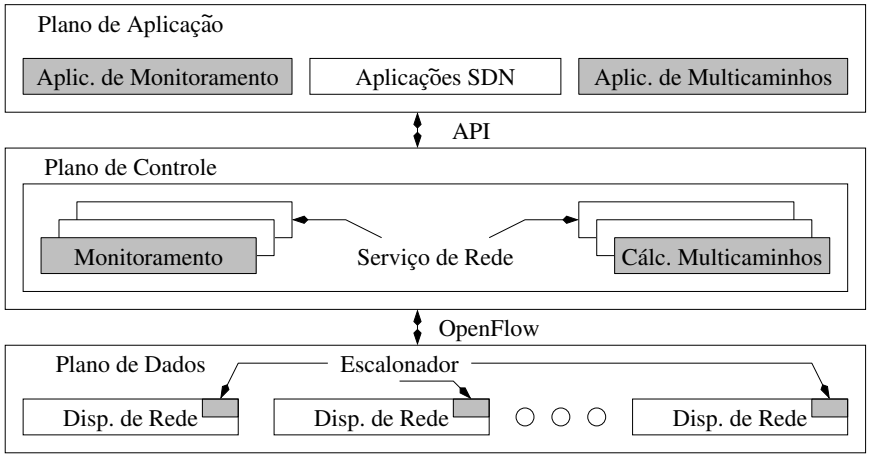


Figura 1. Arquitetura SDN e Adições Propostas.

No Plano de Dados, adiciona-se aos *switches* OpenFlow um escalonador de pacotes, configurado de acordo com decisões do módulo de multicaminhos proposto aqui. O Plano de Controle acomoda dois módulos adicionais. O primeiro, de monitoramento, é responsável pelo processamento e armazenamento de estatísticas dos *switches* [Rezende et al. 2015], e o segundo, Multicaminhos, pelo cômputo de multicaminhos.

De modo similar, o Plano de Aplicação acomoda duas novas aplicações. A primeira, Aplicação de Monitoramento, oferece uma interface para se observar o estado atual de todo o plano de dados, onde informações detalhadas de consumo de recursos são apresentadas em diferentes granularidades [Rezende et al. 2015]. A segunda é a Aplicação de Multicaminhos, onde os usuários podem contratar recursos de rede, indicando a largura de banda desejada para determinado fluxo, ou seja, um *baseline* de Acordo de Nível de Serviço (SLA - na sigla em inglês). Este SLA é a base para o funcionamento do módulo de multicaminhos proposto neste trabalho.

A Figura 2 apresenta maiores detalhes sobre as adições propostas para o plano de controle. Os componentes circundados pela linha pontilhada constituem módulos padrão do Floodlight [Floodlight 2014]. Embora o Floodlight esteja sendo a referência, outros controladores contemplam estruturas similares. A opção pelo Floodlight deu-se por sua natureza de código aberto, modular, com suporte a múltiplas *threads* e suporte completo para o OpenFlow nas versões 1.0 e 1.3. Devido a sua modularidade, a adição de novos módulos é bastante natural e eficiente.

O Módulo de Monitoramento é responsável pela coleta, processamento e armazenamento das estatísticas extraídas dos *switches*. Inicialmente, cabe a este módulo a recuperação da topologia da rede a partir do Módulo de Gerenciamento de Topologia nativo do Floodlight, para na sequência utilizar um mecanismo de coleta, por exemplo por *polling*, para inquirir periodicamente estatísticas nos *switches*. Mais detalhes sobre a operação do módulo de monitoramento podem ser encontrados em [Rezende et al. 2015].

O Módulo de Multicaminhos é responsável pela definição dos caminhos a se-

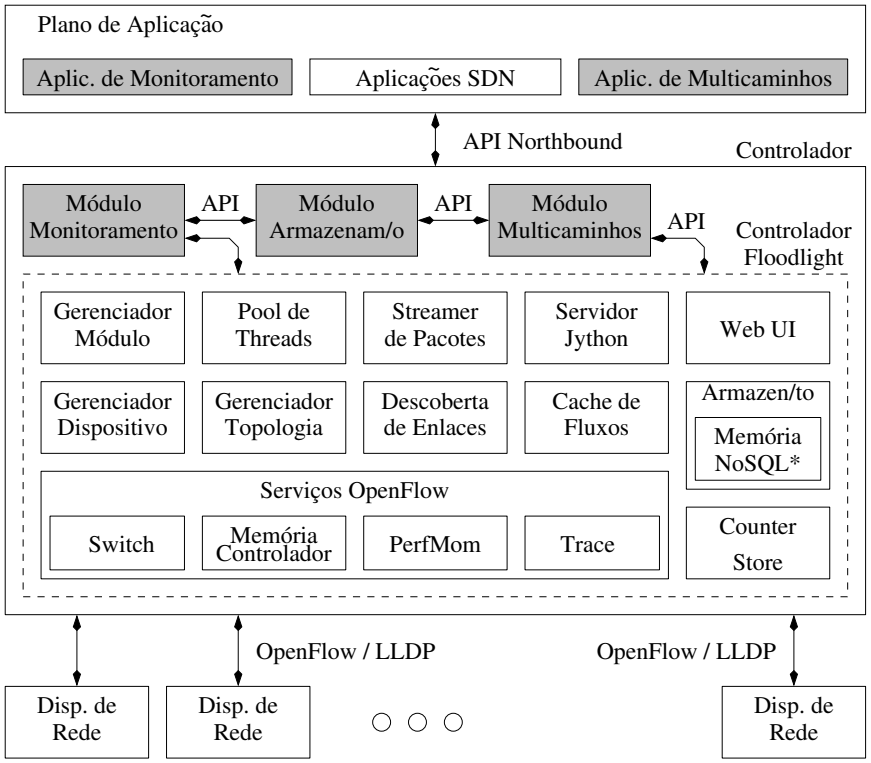


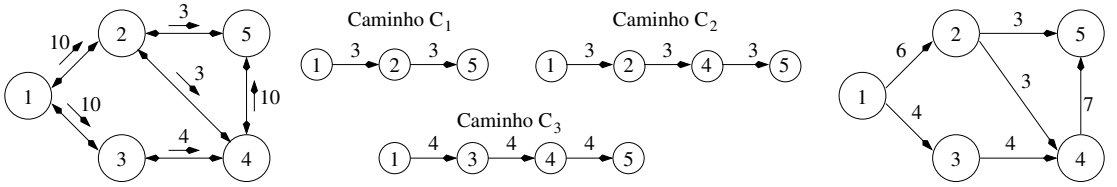
Figura 2. Adições propostas ao Controlador Floodlight.

rem utilizados no encaminhamento de pacotes, de tal forma a garantir o cumprimento de SLAs. Em resumo, a partir da topologia da rede obtida através do Módulo de Gerenciamento de Topologia nativo do Floodlight, considerando o SLA informado pelo usuário através da Aplicação de Multicaminhos e, inquirindo o Módulo de Armazenamento sobre a situação atual do plano de dados que foi gerada pelo Módulo de Monitoramento, o módulo de multicaminhos determina que caminho(s) pode(m) assegurar o SLA e atua no plano de dados. Detalhes do funcionamento deste módulo são apresentados na Seção 3.1.

O Módulo de Armazenamento é um banco de dados de uso geral para os módulos adicionais propostos em nosso trabalho. Nele são armazenados (i) os dados coletados pelo Módulo de Monitoramento sobre a situação atual da rede, em diversas granularidades, (ii) os SLAs acordados com os usuários, informados através da Aplicação de Multicaminhos e, também, (iii) as soluções computadas pelo Módulo de Multicaminhos.

3.1. Módulo de Multicaminhos

A Figura 3 exemplifica o processo de cômputo de multicaminhos. Em todas as fases ilustradas, o tráfego é encaminhado a partir de um *host* localizado atrás do vértice `1` para outro *host* localizado atrás do vértice `5`. O processo inicia-se com a obtenção, por parte do módulo de multicaminhos da topologia (grafo) atual, junto ao Gerenciador de Topologias do Floodlight. Este grafo inicial é enriquecido com informações referentes a largura de banda disponível em cada uma das arestas, conforme ilustrado na Figura 3(a). É importante destacar que os enlaces são bidirecionais, apresentando medidas de largura de banda disponível em ambos os sentidos. Porém, para efeitos de clareza na Figura 3(a), apresentamos apenas as medidas de interesse para a exemplificação em curso.



(a) Grafo inicial e largura de banda disponivel nas arestas.

(b) Conjunto de caminhos disponiveis e respectivas larguras de banda.

(c) Grafo resultante e largura de banda consumida nas arestas.

Figura 3. Exemplo do cômputo de multicaminhos.

A largura de banda disponível é obtida junto ao Módulo de Armazenamento, tendo sido gerada pelo Módulo de Monitoramento. Neste artigo, optamos por considerar apenas a largura de banda, mas o módulo de monitoramento é capaz de obter outras métricas, tais como o atraso experimentado em cada aresta. Todos estes valores poderiam ser utilizados para o cômputo dos multicaminhos, bem como na especificação dos SLAs dos usuários, em abordagem semelhante ao apresentado neste artigo.

A partir do grafo apresentado na Figura 3(a), e considerando o SLA informado pelo usuário, o Módulo Multicaminhos proposto dispara o processo de cálculo de caminhos entre origem e destino. Neste exemplo, consideramos que o usuário solicitou em seu SLA uma taxa fim-a-fim de 10 Mbps, e que os valores expressos em cada uma das arestas estão representados também em Mbps. Utilizando um algoritmo de cálculo de caminho mais curto (e.g. Dijkstra), o Módulo Multicaminhos encontra um caminho disponível entre a origem e destino. Cada caminho é uma sequência de arestas $A_{i,j}$, do vértice i para o j e largura de banda $A_{i,j}.w$.

Ao executar o algoritmo de cálculo de caminho mais curto pela primeira vez, o módulo obtém o Caminho C_1. Na sequência, o Módulo determina a largura de banda deste caminho aplicando a função $LB(C_x) = argmin_{v \in C_x} v.w$. No exemplo da Figura 3(b), $C_1 = \{A_{1,2}, A_{2,5}\}$ e $LB(C_1) = 3$.

Se o valor $LB(C_x)$ for *menor* de que a largura de banda requisitada no SLA, então $LB(C_x)$ é decrementado das larguras de banda das arestas do grafo original que compõem C_x, atualizando o grafo da rede. As arestas que atingirem o valor de largura de banda disponível zero, são removidas do grafo. O valor de $LB(C_x)$ também é decrementado da largura de banda requisitada no SLA.

Todo o processo é repetido até que não existam mais caminhos disponíveis no grafo entre o par origem-destino ou até que o total de banda requisitado seja decrementado do grafo. No primeiro caso, determinou-se que a largura de banda requisitada não pode ser, no momento, atendida, e no segundo determinou-se um conjunto de caminhos que atendem à requisição, $\widehat{CC}$.

Em caso de sucesso na obtenção do conjunto $\widehat{CC}$, o Módulo de Multicaminhos aplicará as regras de encaminhamento necessárias ao plano de controle, utilizando dos recursos de Tabelas de Grupos e *Buckets*, disponíveis no OpenFlow. No exemplo em curso, o SLA de 10 Mbps é atendido através do conjunto $\widehat{CC} = \{C_1, C_2, C_3\}$, ilustrados na Figura 3(b), que permitem a geração do grafo resultante na Figura 3(c), que atende ao SLA solicitado. Cada sentido da comunicação é tratado de maneira independente, sendo possível que o SLA para o sentido *downstream* especifique uma largura de banda

diferente da presente no SLA para o sentido *upstream*. Detalhes da atuação do módulo de multicaminhos no plano de dados serão apresentados na Seção 3.2, em conjunto com detalhamentos das adições propostas para o plano de dados.

Há diversas questões importantes na geração de $\widehat{CC}$ a serem investigadas. Por exemplo, é melhor tentar identificar e usar primeiro caminhos de maior ou menor largura de banda? Com mais ou menos saltos? Quantos caminhos, no máximo? Diversas destas questões, como o fato de algumas das variantes do problema ser NP-Difícil, foram estudadas em outros trabalhos [Martens and Skutella 2006, Duhamel and Chauvet 2003], e a incorporação de tais resultados a este trabalho será objeto de trabalhos futuros.

3.2. Escalonador Multicaminhos

Um *Switch* OpenFlow contém três tipos de tabelas: Tabela de Fluxo; Tabela de Grupo e Tabela de Medidas (*Meter Table*). A Figura 4 descreve o fluxo dos pacotes através de um *Switch* OpenFlow, considerando os três tipos de tabelas disponíveis, bem como posiciona entre os demais elementos o escalonador proposto neste artigo.

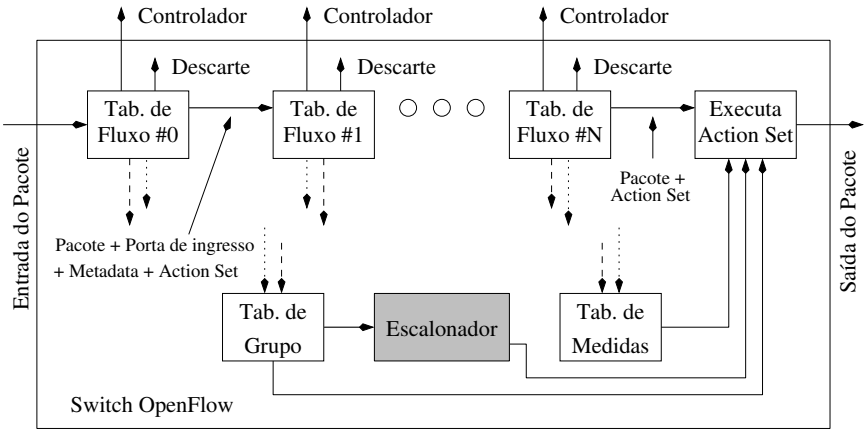


Figura 4. Adições propostas ao Switch OpenFlow.

Como pode ser observado na Figura 4, *switches* OpenFlow contém uma ou mais tabelas de fluxo; quando um pacote é admitido no *switch*, campos do seu cabeçalho são usados para selecionar, em ordem de prioridade, uma entrada da primeira tabela de fluxo, que descreve a ação a ser tomada para o pacote. Se nenhuma entrada for selecionada, o pacote pode ser descartado ou enviado para o controlador.

Quando ocorrer o casamento do cabeçalho com uma entrada da tabela, as instruções associadas com a entrada serão executadas. Estas instruções contém algumas ações que podem encaminhar o pacote para outra tabela de fluxo, modificar o pacote, encaminhá-lo para uma porta lógica ou física, ou mesmo direcioná-lo para um tabela de grupo ou de medidas. Especialmente para o escopo deste trabalho, a proposta é associarmos a ação de encaminhamento dos pacotes de um fluxo para a Tabela de Grupos, onde o Módulo Multicaminhos (detalhado na Seção 3.1) atua, definindo o comportamento do Escalonador proposto, destacado na Figura 4, de tal forma a atender o SLA solicitado pelo usuário, ou seja, aplicando no plano de dados o conjunto $\widehat{CC}$ obtido.

A Tabela de Grupo é constituída por várias entradas, cada qual, permite um método distinto de encaminhamento, como *multicast* ou *broadcast*, usando um ou mais *buc-*

kets de ação para realizar suas necessidades. Se um pacote é enviado para uma entrada de grupo, o pacote será encaminhado de acordo com o tipo de grupo. Existem quatro tipos de grupo: *indirect*, *all*, *select* e *fast failover*. O tipo *indirect* é o mais simples e neste caso o grupo suporta apenas um *bucket*. Normalmente utilizado para apoiar a convergência de forma rápida e mais eficiente. O tipo *all* copia o pacote recebido e o envia para todos os *buckets* no grupo. Normalmente usado para a transmissão e encaminhamento de *multicast*. O tipo *select* permite que os pacotes sejam distribuídos para *buckets* de acordo com um algoritmo de escalonamento. O tipo *fast failover* escolhe o primeiro *bucket* ativo e pode ser utilizado para implementar o *backup* de caminhos.

Nesta proposta, após indexação dos fluxos na Tabela de Grupo, verifica-se se o tipo de grupo é *select* ou não. Se pertencer ao tipo *select*, repassamos o pacote para que o Escalonador proposto possa definir para qual *bucket* encaminhar e este, por sua vez, define através de qual interface física o pacote será enviado. Se não for *select*, o Escalonador proposto é ignorado, e os pacotes são encaminhados conforme padrão OpenFlow.

A Figura 5 detalha o escalonador proposto, baseado na política de escalonamento *round robin*. Neste exemplo, a Tabela de Grupo contém três entradas identificadas como $\#A$, $\#B$ e $\#C$, onde o número do grupo aparece após $G\#$. Já o valor que aparece após $B\#$ representa o número do *bucket* e o número entre parênteses representa o peso do *bucket*. Por fim, o número que segue a letra I representa o número da interface de saída. Neste exemplo, o Grupo $G\#1$ tem dois *buckets*, $B\#1$ e $B\#2$, com pesos 3 e 2, respectivamente. O primeiro *bucket* aponta para a interface $I1$ e o segundo para a interface $I2$. Uma estruturação similar foi definida na Figura 5 para os Grupos $G\#2$ e $G\#3$.

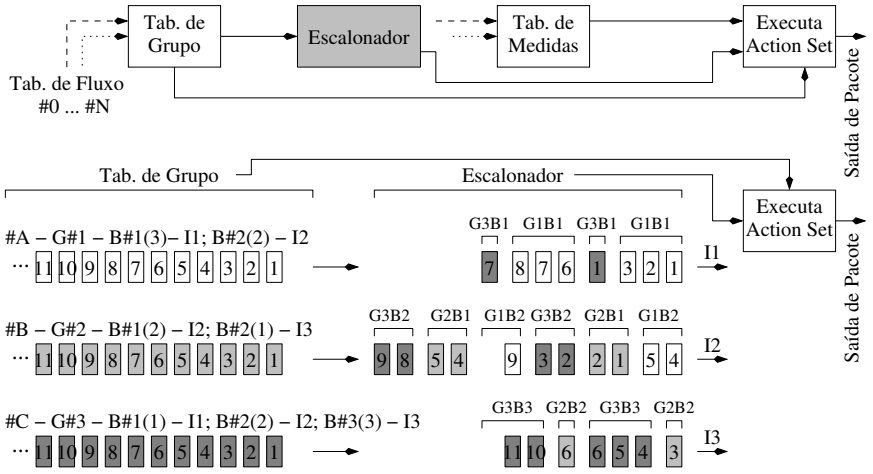


Figura 5. Escalonador - Encaminhamento no Nível do *Bucket* e do Grupo.

O tratamento dispensado para todos os pacotes recebidos em todos os três grupos está ilustrado na Figura 5. No exemplo, o escalonador envia os três primeiros pacotes do $G\#1$ para o *bucket* $B\#1$ e os dois pacotes seguintes para $B\#2$. Os mesmos passos se aplicam para os demais Grupos $G\#2$ e $G\#3$, até que todos os pacotes presentes na figura sejam escalonados.

4. Avaliação Experimental

Nesta seção avalia-se o efeito de múltiplos caminhos, conforme Módulo de Multicaminhos proposto, utilizando um protótipo. Inicialmente descreve-se os cenários usados na

avaliação e, posteriormente, os resultados obtidos.

4.1. Cenários

Duas aplicações distintas, SCP e iPerf foram consideradas na avaliação, ambas com serviço orientado por conexão. Isto é, o TCP é o responsável por prover transferência confiável e ordenada para ambas aplicações. Conforme mencionado anteriormente, o Módulo Multicaminhos é capaz de tratar todos os fluxos que utilizam o plano de dados, sendo a opção pelo SCP e iPerf devido aos dados estatísticos apresentados pelo TCP, essenciais para as análises apresentadas nesta seção.

O uso do iPerf tem por razão a geração de tráfego enquanto se encaminha o mesmo pela rede, já no SCP os dados estão em arquivo, bastando apenas o encaminhamento pela rede. Através do iPerf é possível avaliar quanto se transfere de dados num dado intervalo fixo de tempo (90 segundos), a uma taxa definida a 100 Mbps. Já com o SCP, avalia-se o tempo de transferência de um arquivo de tamanho fixo (1Gb), a uma taxa também definida a 100 Mbps. Desta forma, temos duas aplicações que permitem análises complementares sobre o uso de multicaminhos.

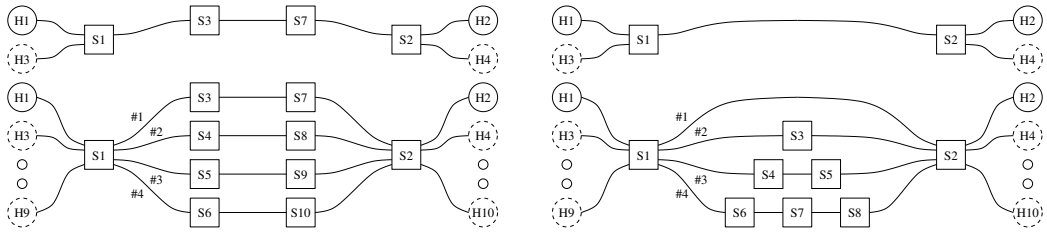
As investigações comparam experimentos em topologias multicaminhos *versus* caminho único, na presença e ausência de tráfego concorrente, isto é, tráfego não gerido pelo módulo de multicaminhos e que concorre com a aplicação avaliada (SCP ou iPerf). Os cenários avaliados são os seguintes:

(i) multicaminhos *versus* caminho único + tráfego concorrente, onde todos os caminhos apresentam o mesmo número de saltos;
(ii) multicaminhos *versus* caminho único, onde todos os caminhos apresentam o mesmo número de saltos;
(iii) multicaminhos *versus* caminho único + tráfego concorrente, onde cada caminho apresenta um número de saltos variado; e,
(iv) multicaminhos, onde cada caminho apresenta um número de saltos variado;

Em todos os cenários, esboçados nas Figuras 6(a) e 6(b), as interfaces de ingresso e egresso dos *switches* e dos *hosts* têm largura de banda de 100 Mbps. Os *hosts* indicados por círculos em linha contínua geram o tráfego para a aplicação avaliada (SCP e iPerf), conforme características definidas anteriormente para tempo, tamanho de arquivo e largura de banda. Os *hosts* indicados por círculos em linha pontilhada geram o tráfego concorrente, que insere carga adicional nos caminhos compondo a topologia. Em todos os cenários, o tráfego concorrente foi gerado utilizando o iPerf, para o protocolo UDP, sendo executado durante todo o tempo do experimento, e sempre a uma taxa de transmissão fixa a 60 Mbps entre cada par de *hosts* origem-destino.

Os experimentos são conduzidos utilizando-se um servidor de *rack* com 02 processadores *quad-core* de 2.4 GHz, 64 GB de memória RAM e HD de 300 GB. Todos os cenários são executados utilizando-se o *VMware ESXi* como *hypervisor*. Todos os *hosts*, *switches* e o controlador são *Virtual Machines* (VMs) assim como também são virtuais os enlaces que os ligam. Utilizou-se o *Open vSwtich* na versão 2.3.2 [Pfaff et al. 2015] como *switch* virtual e o Floodlight 1.0 [Floodlight 2014] como o controlador (Seção 3.2). Cada VM utiliza 1 Processador Virtual, 8 GB de HD e 1 GB de RAM.

A Figura 6(a) apresenta a topologia utilizada na avaliação dos cenários (i) e (ii). Como pode ser observado nesta figura, todos os caminhos são compostos por 5 saltos (4



(a) Multicaminhos com 05 Saltos por Caminho vs Caminho Único.

(b) Multicaminhos com Múltiplos Saltos por Caminho.

Figura 6. Topologias avaliadas.

switches). No caminho único (topo), o tráfego da aplicação avaliada é gerado entre os *hosts* H_1 e H_2, conforme características mencionadas anteriormente, e o tráfego concorrente é gerado a 60 Mbps usando o iPerf entre os *hosts* H_3 e H_4. Para a topologia de multicaminhos (parte inferior da figura), o tráfego da aplicação avaliada também é gerado entre H_1 e H_2, e cada um dos quatro caminhos recebe um tráfego concorrente de 60 Mbps, entre os pares $\{H_3, H_4\}$, $\{H_5, H_6\}$, $\{H_7, H_8\}$ e $\{H_9, H_{10}\}$.

A Figura 6(b) contempla múltiplos caminhos entre fonte e destino, onde o número de saltos varia por caminho, ou seja, 03 saltos pelo caminho #1 até 06 saltos pelo caminho #4, incluindo caminhos menores e maiores (em número de saltos) quando comparados com a topologia simétrica. Esta figura é usada nas avaliações referentes aos cenários (iii) e (iv). De maneira equivalente aos cenários anteriores, no caminho único (topo), o tráfego da aplicação avaliada é gerado entre os *hosts* H_1 e H_2, e o tráfego concorrente é gerado entre os *hosts* H_3 e H_4. Para a topologia de multicaminhos (parte inferior), o tráfego da aplicação avaliada também é gerado entre H_1 e H_2, e cada um dos quatro caminhos recebe um tráfego concorrente entre os pares $\{H_3, H_4\}$, $\{H_5, H_6\}$, $\{H_7, H_8\}$ e $\{H_9, H_{10}\}$.

4.2. Resultados

A Tabela 1 detalha o volume total de dados transferidos entre os *hosts* em MBytes, a taxa de transmissão experimentada em Mbps, o tempo total para a transferência em segundos e dados sobre o total de pacotes e total de retransmissões do TCP. Esta tabela refere-se aos experimentos do cenário (i), e os valores apresentados, são a média dos valores coletados em 5 execuções de cada experimento. Em todos os experimentos apresentados nesta seção o desvio padrão observado foi extremamente baixo, por exemplo, o maior valor identificado foi na ordem de 9,065E-005 para o cenário (i), justificando a opção por 5 execuções em cada caso. Nas avaliações considerando o uso de multicaminhos, o escalonador foi ajustado com uma taxa de 25% do tráfego encaminhado através de cada um dos quatro caminhos disponíveis na topologia da Figura 6(a).

Observa-se que para o iPerf, o multicaminhos possibilita a transferência de 1022 MB em 90 segundos, enquanto que no caminho único, apenas 416 MB são enviados no mesmo período. Há ganho explícito de vazão (largura de banda) na rede, experimentando 38,7 Mbps pelo caminho único e atingindo 95,22 Mpbs ao distribuir o mesmo tráfego entre os múltiplos caminhos. Ainda referente à análise do iPerf, observa-se que o caminho único, por estar saturado entre a aplicação avaliada e o tráfego concorrente, leva a um total de 0,1538% de pacotes descartados, e este volume de pacotes descartados é reduzido para 0,0512% ao utilizarmos os multicaminhos, onde os caminhos não estão saturados.

Tabela 1. iPerf/SCP - Multicaminhos com 05 Saltos/Caminho + Tráfego de Fundo.

Dados (MBytes)	Vazão (Mbps)	Tempo (seg)	SEGs Enviados	SEGs* REXMIT	SEGs(%) REXMIT
iPerf - Caminho Único + Tráfego de Fundo					
416,00	38,70	90,00	300929	382;00;00	0,1538%
iPerf - Multicaminhos com 05 Saltos + Tráfego de Fundo					
1022,00	95,22	90,00	739896	374;04;01	0,0512%
SCP - Caminho Único + Tráfego de Fundo					
1024,00	36,80	222,00	742458	855;00;00	0,1150%
SCP - Multicaminhos com 05 Saltos + Tráfego de Fundo					
1024,00	92,36	93,00	742647	409;24;01	0,0584%

* REXMIT = SEGs Retransmitidos [Fast; Forward; Slow Start].

Os resultados apresentados na avaliação do SCP, onde o tamanho do arquivo a ser transferido é fixo em 1GB (1024MB), observa-se que o uso de um caminho único, saturado pela existência do tráfego concorrente, eleva o tempo total de transmissão para 222 segundos, ao passo que ao recorrermos ao uso de multicaminhos, este tempo total de transmissão é reduzido para 93 segundos. Esta variação no tempo total, também gera uma variação na taxa média de transmissão, saindo de 36,80 Mbps de taxa efetiva no caminho único, para 92,36 Mbps no multicaminhos. Os resultados referentes ao volume de retransmissões indicam 0,1150% para o caminho único e este valor cai para 0,0584% ao utilizarmos o multicaminhos.

Ainda para o cenário da Figura 6(a), as Tabelas 2 (a) e (b) detalham os dados coletados na ausência de tráfego de fundo. Estes experimentos representam o cenário (ii) e possuem o objetivo de avaliar qual o impacto do uso de multicaminhos, quando na verdade um único caminho seria capaz de atender o SLA solicitado. Desta forma, definimos quatro conjuntos de pesos distintos para o escalonador: (a) {85%, 5%, 5%, 5%}; (b) {65%, 20%, 10%, 5%}; (c) {45%, 30%, 15%, 10%}; e (d) {25%, 25%, 25%, 25%} . Os dados apresentados referem-se a média dos valores coletados de 05 execuções em caminho único e 05 execuções para cada um dos conjuntos de pesos para o caso multicaminhos, totalizando 25 execuções.

Tabela 2. iPerf / SCP - Caminho Único vs Multicaminhos com 05 Saltos/Caminho.

(a) iPerf.

Dados (MBytes)	Vazão (Mbps)	Tempo (seg)	SEGs Enviados	SEGs* REXMIT	SEGs(%) REXMIT
Caso #1 - #1=100					
1024,00	95,62	90,00	743024	122;00;01	0,0166%
Caso #2 - #1=85; #2=05; #3=05; #4=05					
1022,00	95,22	90,00	740005	377;06;01	0,0519%
Caso #3 - #1=65; #2=20; #3=10; #4=05					
1023,00	95,34	90,00	740945	376;02;01	0,0511%
Caso #4 - #1=45; #2=30; #3=15; #4=10					
1024,00	95,50	90,00	742394	386;01;01	0,0522%
Caso #5 - #1=25; #2=25; #3=25; #4=25					
1024,00	95,30	90,00	740710	410;04;01	0,0560%

(b) SCP.

Dados (MBytes)	Vazão (Mbps)	Tempo (seg)	SEGs Enviados	SEGs* REXMIT	SEGs(%) REXMIT
Caso #1 - #1=100					
1024,00	92,36	93,00	742579	150;00;01	0,0203%
Caso #2 - #1=85; #2=05; #3=05; #4=05					
1024,00	92,36	93,00	742619	393;11;01	0,0546%
Caso #3 - #1=65; #2=20; #3=10; #4=05					
1024,00	92,36	93,00	742581	397;18;01	0,0560%
Caso #4 - #1=45; #2=30; #3=15; #4=10					
1024,00	92,36	93,00	742616	404;19;01	0,0571%
Caso #5 - #1=25; #2=25; #3=25; #4=25					
1024,00	92,36	93,00	742584	406;19;01	0,0573%

Como observado na Tabela 2 (a), o iPerf consegue gerar praticamente o mesmo volume de tráfego dentro da janela de 90 segundos, resultando em uma taxa de transmissão efetiva praticamente estável, independente do fato de usarmos caminho único ou diferentes pesos para multicamihos. Entretanto, é possível observar que o uso de multicaminhos gera uma elevação no número de retransmissões, variando entre 3,08 a 3,37 vezes o número de retransmissões identificados no caminho único.

O mesmo tipo de comportamento é observado para o SCP na Tabela 2 (b), onde o tempo total para a transmissão do mesmo arquivo de 1024MB é estável, frente ao uso

de qualquer configuração. O número de retransmissões no multicaminhos varia de 2,69 a 2,82 vezes o número no caminho único, um valor ligeiramente melhor do que os apresentados nas avaliações para o iPerf (Tabela 2 (a)). A variação observada nas retransmissões quando utilizados os múltiplos caminhos, entre os diferentes pesos, é bastante sutil, sendo praticamente estáveis. Porém, indicam que o uso de multicaminhos leva a mais retransmissões, quando comparado ao uso de caminho único.

Basicamente, ao segmentarmos o fluxo em quatro sub-fluxos, o TCP torna-se mais suscetível ao envio de mensagens de reconhecimento (ACKs) duplicados, pois os segmentos chegam fora de ordem ao destinatário. Como pode ser observado, a maior taxa de retransmissão refere-se a *Fast Retransmissions*, que ocorrem quando o contador de ACKs duplicados atinge o total de três ACKs com mesmo número de reconhecimento. Ou seja, os resultados indicam que na existência de um caminho único capaz de honrar o SLA, a solução multicaminhos não é a mais indicada.

Os resultados apresentados na Tabela 3 correspondem ao cenário (iii). Da mesma forma que no cenário (i), os resultados evidenciam o ganho explícito de vazão na rede para o iPerf e de tempo de transferência do arquivo para o SCP. Em ambos os casos, há redução significativa do número de retransmissões TCP, ou seja, 0,1427% para 0,0527% para o iPerf e 0,1275% para 0,0554% para o SCP. Todos os valores são equivalentes aos valores observados no cenário (i), detalhados na Tabela 1. Estes resultados demonstram novamente que a solução de multicaminhos, em condições nas quais esta divisão de tráfego faz-se necessária, melhoram todos os aspectos da transmissão. Ou seja, quando o plano de dados não apresenta um único caminho capaz de atender integralmente o SLA solicitado pelo usuário, o uso de multicaminhos é essencial para garantir o SLA.

Tabela 3. Multicaminhos com 03 a 06 Saltos/Caminho + Tráfego de Fundo.

Dados (MBytes)	Vazão (Mbps)	Tempo (seg)	SEGs Enviados	SEGs* REXMIT	SEGs(%) REXMIT
iPerf - Caminho Único + Tráfego de Fundo					
416,00	38,68	90,00	300894	430;00;00	0,1427%
iPerf - Multicaminhos com 03 a 06 Saltos + Tráfego de Fundo					
1023,00	95,38	90,00	741256	387;02;01	0,0527%
SCP - Caminho Único + Tráfego de Fundo					
1024,00	38,59	222,00	742448	948;00;00	0,1275%
SCP - Multicaminhos com 03 a 06 Saltos + Tráfego de Fundo					
1024,00	92,36	93,00	742636	394;17;01	0,0554%

As Tabelas 4 (a) e (b) detalham os dados coletados para o iPerf e SCP no cenário (iv), onde os quatro conjuntos de pesos descritos anteriormente são utilizados. Neste caso, diferentemente do observado no cenário (ii), onde a topologia é toda simétrica em número de saltos, os resultados indicam certa variação referente aos número de retransmissões observadas, especialmente no Caso #1, quando o percentual de 85% de todo o tráfego é encaminhado através do caminho com menor número de saltos.

Os resultados deste cenário (iv), indicam ainda redução em número de retransmissões, frente ao observado nos resultados do cenário (ii), justificados pelo número um pouco menor de saltos totais. No cenário (ii), o tráfego entre H_1 e H_2 utiliza quatro caminhos com 4 *switches* em cada, em um total de 16 *switches*. No cenário (iv), o total de *switches* entre H_1 e H_2 cai para 14.

Em uma análise de RTT entre H_1 e H_2 na Figura 6(b) podemos observar o *overhead* introduzido pelo protótipo a cada novo *switch* + enlace inseridos nas topologias.

Tabela 4. iPerf/SCP - Multicaminhos com 03 a 06 Saltos/Caminho.

(a) iPerf.

Dados (MBytes)	Vazão (Mbps)	Tempo (seg)	SEGs Enviados	SEGs* REXMIT	SEGs(%) REXMIT
Caso #1 - #1=85; #2=05; #3=05; #4=05					
1023,00	95,38	90,00	741270	285;04;01	0,0359%
Caso #2 - #1=65; #2=20; #3=10; #4=05					
1023,00	95,36	90,00	741163	323;02;01	0,0442%
Caso #3 - #1=45; #2=30; #3=15; #4=10					
1024,00	95,58	90,00	742774	321;02;01	0,0436%
Caso #4 - #1=25; #2=25; #3=25; #4=25					
1024,00	95,24	90,00	742448	341;02;01	0,0463%

(b) SCP.

Dados (MBytes)	Vazão (Mbps)	Tempo (seg)	SEGs Enviados	SEGs* REXMIT	SEGs(%) REXMIT
Caso #1 - #1=85; #2=05; #3=05; #4=05					
1024,00	92,36	93,00	742602	288;22;01	0,0419%
Caso #2 - #1=65; #2=20; #3=10; #4=05					
1024,00	92,36	93,00	742629	352;23;01	0,0506%
Caso #3 - #1=45; #2=30; #3=15; #4=10					
1024,00	92,36	93,00	742584	319;07;01	0,0440%
Caso #4 - #1=25; #2=25; #3=25; #4=25					
1024,00	92,36	93,00	742599	318;05;01	0,0436%

Em uma média de 100 pings entre H_1 e H_2, utilizando cada um dos caminhos ilustrados na Figura 6(b) temos: #1 - 0,469ms; #2 - 0,646ms; #3 - 0,763ms; e #4 - 0,879ms.

Embora o SLA adotado neste artigo, para desenvolvimento do Módulo de Multicaminhos, considere apenas o requisito de largura de banda, os resultados das Tabelas 4(a) e (b), indicam que variações no número de saltos e, também, variações de atraso possuem participação no comportamento de multicaminhos. Como pode ser observado pelos valores médios de RTT, o Caminho #1 apresentou o menor dos atrasos. Os resultados indicam ainda que a adição de novos *switches* aos caminhos vai aumentando de maneira similar ($\approx 0,100ms$) o RTT experimentado. Ao atribuir 85% do tráfego pelo caminho 1, uma parcela significativa dos pacotes tende a chegar de maneira ordenada ao destinatário, aliviando os efeitos observados nos demais casos. Estes resultados são importantes, pois indicam como os trabalhos futuros devem atuar na definição de novas heurísticas para o cômputo do conjunto $\widehat{CC}$ pelo módulo de multicaminhos.

5. Conclusões e Trabalhos Futuros

Neste artigo, apresentamos proposta de módulos adicionais à Arquitetura SDN OpenFlow para roteamento multicaminhos, como forma de atender demandas de nível de serviço da aplicação. No contexto de aplicações que demandam níveis estritos de SLA, onde os efeitos de congestionamentos devem apresentar um baixo reflexo para a aplicação, as extensões propostas em nosso trabalho, em todos os níveis da Arquitetura SDN, mostraram-se eficazes.

Ao incorporarmos uma visão precisa e enriquecida do plano de dados, através do Módulo de Monitoramento [Rezende et al. 2015], o Módulo de Multicaminhos apresentado e avaliado neste artigo foi capaz de atender a largura de banda requisitada, bem como proporcionou baixo percentual de retransmissões TCP, quando comparado com o caso de caminho único sob as mesmas condições de nível de serviço.

De maneira geral, as aplicações demandam cada vez mais largura de banda e desempenho da rede, em um cenário onde novas aplicações aparecem a todo instante. Ao ofertarmos uma solução desenvolvida integralmente em nível da SDN, onde multicaminhos são gerados para qualquer tipo de fluxo de dados sem a percepção por parte das aplicações, ampliamos a gama de cenários que podem se beneficiar da solução proposta. O Módulo Multicaminhos é capaz de aglutinar fatias de largura de banda para atender a um dado nível de serviço, maximizando o uso dos recursos de rede.

Outras questões, tais como quais caminhos, com quais características, devem ser escolhidos dentre aqueles que satisfaçam um dado nível de serviço; tendo por base o número de saltos, variações de atraso, número máximo de caminhos e, outros parâmetros,

constituem questões com elevado grau de complexidade, que necessitam de mais pesquisa e, por isso, constituem tópicos para futuros trabalhos.

Agradecimentos

Os autores gostariam de agradecer ao suporte da CAPES, CNPq e Fapemig na realização deste trabalho.

Referências

Duhamel, B. Vatinlen, P. M. and Chauvet, F. (2003). Minimizing congestion with a bounded number of paths. In *Proceedings of Algorithmic of Telecommunications (ALGOTEL '03*, pages 155–160.

Floodlight, P. (2014). `http://www.projectfloodlight.org`.

Ford, A., Raiciu, C., Handley, M., and Bonaventure, O. (2013). TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, IETF. ISSN 2070-1721.

Foundation, O. N. (2013). OpenFlow Switch Specification - Version 1.3.2.

Jinyao, Y., Hailong, Z., Qianjun, S., Bo, L., and Xiao, G. (2015). Hiqos: An sdn-based multipath qos solution. *Communications, China*, 12(5):123–133.

Li, Y. and Pan, D. (2013). Openflow based load balancing for fat-tree networks with multipath support. In *Proc. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary*, pages 1–5.

Martens, M. and Skutella, M. (2006). Flows on few paths: Algorithms and lower bounds. *Networks*, 48(2):68–76.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., and Casado, M. (2015). The Design and Implementation of Open vSwitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 117–130, Oakland, CA. USENIX Association.

Rezende, P. H., Coelho, P. R., Faina, L. F., Camargos, L., and Pasquini, R. (2015). Plataforma para Monitoramento de Métrica de Nível de Serviço em Redes Definidas por Software. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Vitoria, ES, Brazil.

Rezende, P. H. A. (2016). Extensões na Arquitetura SDN para o Provisionamento de QoS Através do Monitoramento e Uso de Múltiplos Caminhos. *Dissertação de Mestrado apresentada ao PPGCO/FACOM/UFU*.

Sandri, M., Silva, A., Rocha, L., and Verdi, F. (2015). On the benefits of using multipath tcp and openflow in shared bottlenecks. In *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*, pages 9–16.

Thaler, D. (2000). Multipath issues in unicast and multicast next-hop selection. internet engineering task force: Rfc 2991.

van der Pol, R., Boele, S., Dijkstra, F., Barczyk, A., van Malenstein, G., Chen, J. H., and Mambretti, J. (2012). Multipathing with mptcp and openflow. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 1617–1624. IEEE.

Contando os Segundos: Avaliação de Estratégias de Domínio Temporal para a Gerência de Regras em Redes SDN

Miguel Neves, Rodrigo Oliveira, Fabrício Mazzola, Daniel Marcon, Luciano Gaspary, Marinho Barcellos

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{mcneves,ruas.oliveira,fmmazzola,dsmarcon,paschoal,marinho}@inf.ufrgs.br

Abstract. *Powered by Software Defined Networks (SDN), network administrators can achieve better control over traffic. This advantage comes at the cost of increased number of rules, which, in turn, leads to resource scarcity and lower performance. Recent proposals attempt to make better use of flow tables by optimizing the time each rule remains installed. In this paper, we perform an in-depth analysis of state-of-the-art strategies. First, we identify their principles and characterize their design space. Then, we perform an extensive experimental evaluation to show their benefits and limitations. Results indicate that the performance of current strategies is far from optimal and heavily relies on a precise parameterization. Our findings provide the foundations for the development of robust strategies to address the limitations of the literature.*

Resumo. *As Redes Definidas por Software (SDN) permitem aos administradores de rede um melhor controle sobre o tráfego de dados. Entretanto, tal vantagem requer um aumento do número de regras de encaminhamento a serem instaladas nas tabelas de fluxos de switches e roteadores, acarretando escassez de recursos e menor desempenho. Propostas recentes tentam fazer melhor uso das tabelas através da otimização do tempo que cada regra permanece instalada. Neste artigo, apresentamos uma avaliação qualitativa e quantitativa abrangente sobre as principais estratégias presentes no estado-da-arte. Primeiramente, identificamos os seus princípios e determinamos o espaço de projeto associado. Em seguida, realizamos extensas avaliações experimentais a fim de mostrar os seus benefícios e limitações. Resultados indicam que o desempenho dos métodos atuais está distante do comportamento ótimo e também depende fortemente de uma parametrização precisa. As conclusões obtidas servem de base para o desenvolvimento de estratégias mais robustas, que possam tratar as limitações encontradas na literatura.*

1. Introdução

A habilidade de gerenciar fluxos com granularidade fina é um dos principais benefícios das Redes Definidas por Software (SDN). Por exemplo, um provedor de infraestrutura de nuvem pode facilmente implementar mecanismos de reserva de banda entre pares de máquinas virtuais (VMs) ou um administrador de rede pode coletar estatísticas em nível de fluxo de forma eficiente utilizando dispositivos SDN [Moshref et al. 2013, Su et al. 2015]. Contudo, a possibilidade de gerenciar o tráfego da rede com tal granularidade possui custos associados, explicados a seguir.

Em redes SDN, políticas são expressas através de regras de encaminhamento, e instaladas em tabelas de fluxos nos dispositivos por meio de um controlador. Tais regras podem ter um tamanho considerável (até 44 campos de cabeçalho em versões recentes do protocolo OpenFlow [ONF 2015]), e ainda ocupar espaço de armazenamento e capacidade de processamento extras para a manutenção de contadores associados [Moshref et al. 2014]. Além disso, aliado à tendência crescente no tamanho das regras, os dispositivos SDN suportam uma quantidade baixa de entradas nas suas tabelas de fluxos, tipicamente uma ou duas ordens de grandeza menores do que o necessário [Katta et al. 2014]. Nesse contexto, o gerenciamento adequado das regras de encaminhamento é uma exigência.

O uso de mecanismos de gerência baseados em agregação (através de *wildcards*) ou posicionamento de regras, tipicamente empregados como solução, trazem a desvantagem de ofuscar informações importantes sobre os fluxos e aplicar desvios muitas vezes indesejados no caminho dos pacotes, respectivamente [Shirali-Shahreza and Ganjali 2015, Nguyen et al. 2015]. Por essa razão, propostas recentes tem buscado melhorar a eficiência no uso das tabelas de fluxos através de estratégias de gerência baseadas no tempo de duração das regras (ou seja, no tempo que as mesmas permanecem instaladas) [Zhang et al. 2014, Vishnoi et al. 2014, Kannan and Banerjee 2014]. Esse conjunto de propostas (definidas como sendo de domínio temporal), no entanto, carece de um estudo comparativo abrangente, capaz de delinear o espaço de projeto envolvido, determinar o desempenho relativo entre as estratégias e apontar direções relevantes de pesquisa.

Neste artigo, busca-se analisar em profundidade as principais propostas de domínio temporal para a gerência de regras e avaliar o impacto que as mesmas acarretam no desempenho de redes SDN. Os principais resultados obtidos indicam que: (i) as estratégias apresentam uma sobrecarga de utilização das tabelas de fluxos em torno de 15% em cenários de superprovisionamento de recursos; (ii) as estratégias são pelo menos 2x piores que o caso ótimo com relação ao número de intervenções do controlador em cenários de escassez de recursos; (iii) pequenas mudanças nas características do tráfego afetam consideravelmente o desempenho das estratégias; e (iv) o uso de mecanismos de timeout (sem um controle adequado) como ferramenta para a gerência de regras pode ser ruim na prática (ao contrário do que a literatura indica).

As principais contribuições deste artigo são: (i) o mapeamento do espaço de projeto e a análise das propriedades essenciais de cada estratégia; (ii) a avaliação comparativa do desempenho das estratégias considerando uma carga de trabalho representativa; e (iii) a identificação de direções relevantes de pesquisa, com base em resultados obtidos experimentalmente e de forma a proporcionar o desenvolvimento de novos estudos. Este é o primeiro trabalho a fazer uma análise sistemática e abrangente da área.

O restante do artigo está organizado da seguinte maneira. A Seção 2 introduz as principais propriedades que determinam o espaço de projeto das estratégias de gerenciamento temporal de regras, enquanto a Seção 3 apresenta modelos formais e discute as principais estratégias encontradas na literatura. A Seção 4, por sua vez, descreve a metodologia utilizada na avaliação de desempenho e os resultados obtidos. Finalmente, a Seção 5 discute trabalhos relacionados e a Seção 6 expõe as principais conclusões do estudo e perspectivas de trabalhos futuros.

2. Propriedades das Estratégias de Gerenciamento Temporal de Regras

As estratégias de gerenciamento temporal de regras podem ser caracterizadas conforme três propriedades centrais: a natureza do mecanismo de contagem da duração das regras; o grau de conhecimento que as estratégias detêm sobre o estado dos fluxos; e o algoritmo de cache aplicado para escolher uma regra a ser eliminada quando a tabela de fluxos se encontra cheia.

Natureza do tempo de duração (timeout) das regras. Grande parte das implementações de SDN suporta a eliminação automática de uma regra após o esgotamento do seu tempo de duração. Nesse sentido, as estratégias precisam considerar tanto o valor do timeout quanto o tipo de contador utilizado para controlá-lo, como explicado a seguir.

O valor do timeout pode ser definido *estaticamente* ou *dinamicamente*. No modo estático, um valor fixo é determinado previamente e utilizado em todas as regras instaladas. Esse valor não sofre variações e é aplicado mesmo se a duração dos fluxos mudar amplamente ao longo do tempo. Em contraste, no modo dinâmico, cada regra pode apresentar um valor diferente de timeout. Nesse caso, o timeout pode ser ajustado ao longo do tempo levando-se em conta o comportamento do fluxo associado (por exemplo, em relação ao tipo, tamanho e frequência de pacotes).

Existem dois tipos de contadores sobre a duração das regras: *por inatividade (idle)* e *rígido (hard)*. No modo inativo, a duração da regra é reajustada para o valor original sempre que acontece uma associação com pacotes. Em outras palavras, a entrada para o fluxo somente será removida se não acontecer nenhuma associação com pacotes durante a duração determinada para a regra. Em contraste, timeouts no modo rígido provocam a eliminação das regras caso as suas respectivas durações sejam transcorridas, independentemente das associações ocorridas.

Conhecimento sobre o estado dos fluxos. Algumas das características dos fluxos (incluindo duração, número de *bytes* e frequência de pacotes) podem afetar amplamente o desempenho dos mecanismos de timeout. Desse modo, um maior conhecimento sobre o estado de um dado fluxo é benéfico, pois decisões mais informadas podem ser feitas sobre a duração das regras. No entanto, a obtenção dessas informações pode ser custosa para a rede, uma vez que requer um processamento adicional nos nodos (switches e controlador) e normalmente gera uma troca extra de mensagens de controle. As estratégias costumam diferir amplamente em relação ao tipo e a quantidade de estado coletada da rede, o que é discutido na Seção 3.

Algoritmos de Cache de Regras. Tais algoritmos determinam qual será o comportamento da estratégia no momento em que a tabela estiver totalmente ocupada. Neste estado operacional, antes que uma nova regra seja instalada, o algoritmo de cache precisa eliminar uma entrada da tabela. Esses algoritmos possuem diferentes requisitos de processamento e sobrecargas, o que significa que a sua eficiência depende das condições operacionais da rede. Os algoritmos adotados pelas estratégias serão discutidos na Seção 3, onde as mesmas são formalmente modeladas e explicadas.

3. Modelagem das Estratégias

Esta seção visa modelar formalmente e discutir as principais estratégias encontradas na literatura, a fim de possibilitar uma comparação justa e precisa entre elas. Para auxiliar

nesse processo, a Tabela 1 resume os parâmetros usados em cada estratégia, enquanto a Tabela 2 mostra as suas informações mais relevantes.

Timeout Estático (ST). Essa estratégia representa o método padrão. A ideia principal consiste em encontrar um valor genérico (e estático) adequado para o tempo de duração de uma regra e aplicá-lo a todas as regras. Nesse caso, timeouts por inatividade são preferidos, pois evitam a eliminação de regras durante uma rajada de pacotes. Essa estratégia utiliza um único parâmetro (o valor estático do timeout – t_{ST}) e não possui nenhum algoritmo de cache associado. Assim como em [Vishnoi et al. 2014], uma eliminação aleatória foi considerada para situações nas quais a tabela está totalmente ocupada, mantendo a simplicidade da estratégia.

Remoção antecipada probabilística (probabilistic early eviction - PEV). Essa estratégia tenta eliminar regras inativas das tabelas de fluxos antes que os seus respectivos tempos de duração expirem. A eliminação é feita mesmo que a tabela não esteja cheia, visando remover entradas antes que os algoritmos de cache sejam acionados. Se ainda assim a tabela fica totalmente ocupada (por exemplo, devido à taxa de chegada de fluxos ser maior que a taxa de eliminação de regras), então o algoritmo LRU (*Least Recently Used*) é aplicado. Essa estratégia é composta por três ações: (i) determinação de um timeout - t_{PEV} (longo, estático e por inatividade) - para as regras; (ii) monitoramento do estado dos fluxos; e (iii) eliminação periódica (a cada DI unidades de tempo) de regras desnecessárias, a partir da probabilidade $P(i)$ do fluxo i ter encerrado.

Existem pelo menos duas maneiras diferentes de se determinar o melhor momento para a eliminação de uma regra (ou seja, $P(i)$). Em [Zarek 2012], um switch *OpenFlow* é estendido para procurar pacotes *SYN* e *FIN* em conexões TCP. Em [Kannan and Banerjee 2014], por sua vez, o controlador periodicamente tenta eliminar regras calculando $P(i)$ a partir das durações dos fluxos e dos intervalos entre pacotes monitorados na rede. Nosso modelo de PEV adota o método em [Kannan and Banerjee 2014], já que o mesmo independe de protocolos específicos, aplicando-se tanto a tráfego TCP como UDP.

Incremento Adaptativo (AI). Proposta por [Vishnoi et al. 2014], possui o objetivo de manter a utilização da tabela de fluxos baixa, administrando sob demanda fluxos longos. Essa estratégia é composta por dois passos: (i) estabelecer um timeout curto (t_{AI}^{min}) adequado para a maioria das regras; e (ii) atualizar a duração de uma regra a cada reinstalação até um valor máximo (t_{AI}^{max}), usando uma função de aumento $f(n\_reinstall)$. A função de aumento considera o número total de reinstalações ($n\_reinstall$) de cada regra, sendo o controlador responsável por manter essa informação. Nosso modelo de AI, conforme sugerido pelos autores em [Vishnoi et al. 2014], adota um algoritmo aleatório de cache para eliminar regras quando a tabela de fluxos estiver cheia.

Timeout rígido adaptativo (Adaptive hard timeout - AHT). Proposta por [Zhang et al. 2014], essa estratégia determina dinamicamente uma duração única para todas as regras levando em conta características do tráfego, capacidades das tabelas de fluxos e restrições de desempenho da rede (por exemplo, tempo máximo de conclusão dos fluxos ou taxa máxima de intervenções do controlador). O objetivo é ajustar o tempo de vida das regras de acordo com as condições operacionais da infraestrutura. De forma geral, a estratégia é composta por três passos: (i) estimar a carga da rede como uma função

Parâmetro	Descrição
Timeout Estático (ST)	
t_{ST}	Timeout definido na instalação das regras
Remoção Antecipada Probabilística (PEV)	
t_{PEV}	Timeout definido na instalação das regras
DI	Intervalo periódico para tentar eliminar uma regra
$P(i)$	Probabilidade de eliminar uma regra inativa
Incremento Adaptativo (AI)	
t_{AI}^{min}	Timeout mínimo - Valor inicial para o timeout
t_{AI}^{max}	Timeout máximo - Limite superior para o timeout
$f(n\_reinstall)$	Função de incremento do timeout
Timeout Rígido Adaptativo (AHT)	
λ	Taxa estimada de chegada de fluxos
μ	Duração estimada dos fluxos
$C = c_1, c_2, ..., c_n$	Conjunto de restrições de desempenho
$t_{AHT}(\lambda, \mu, R, [C])$	Timeout como uma função da carga da rede (λ, μ), do tamanho da tabela (R) e de restrições de desempenho ($[C]$)

Tabela 1. Lista de parâmetros usados em cada estratégia.

composta tanto pela taxa de chegada (λ) quanto pela duração (μ) dos fluxos; (ii) verificar se o valor atual estipulado para a duração das regras (t_{AHT}) satisfaz as restrições de desempenho $(c1, c2, ..., cn)$ especificadas pelo administrador da rede; e (iii) se necessário, atualizar t_{AHT} de forma a atender todas as restrições.

A proposta de [Zhang et al. 2014] assume ausência de algoritmo de cache, em que regras serão removidas somente quando os seus respectivos tempos esgotarem. Essa premissa implica que novos fluxos deverão ser tratados diretamente pelo controlador até que alguma regra expire. Além disso, a mesma (premissa) pode ou não ser acompanhada por uma restrição de sobrecarga máxima do plano de controle (isto é, da taxa de intervenções do controlador). Caso seja, a duração das regras tende a ser gradualmente reduzida quando a tabela fica cheia, de forma a multiplexar o uso de suas entradas entre os fluxos ativos. Essa estratégia é baseada em timeouts rígidos, visto que tais timeouts refletem diretamente o tempo de duração dos fluxos.

Timeout ótimo (OPT). Para fins de comparação entre as estratégias, é descrita uma solução que minimiza o número de intervenções do controlador (ou seja, de mensagens de controle) na rede. Essa solução baseia-se na premissa de que informações precisas sobre o instante de chegada, o intervalo entre pacotes e a duração dos fluxos estariam disponíveis. Caso essas informações estivessem disponíveis, seria possível tratar o problema de determinar a duração das regras como um problema de escalonamento do uso da tabela de fluxos. Nesse sentido, uma regra pode ser removida da tabela assim que possível, reduzindo também o número de entradas ocupadas. A estratégia associada a essa solução opera da seguinte maneira: (i) realização da leitura de informações sobre os fluxos; e (ii) instalação de cada regra no momento em que o respectivo fluxo inicia, com duração idêntica à deste. Quando a tabela fica cheia, a regra referente ao fluxo cujo próximo pacote está mais distante no futuro é removida.

4. Avaliação

Nesta seção, apresentamos a avaliação realizada considerando diferentes cenários de operação da rede (p.ex. sobrecarga, tráfego variável). Entre os principais objetivos, está responder às seguintes questões: (i) qual a melhor estratégia de gerência temporal de regras para cada cenário? (ii) quais as melhores escolhas de projeto (por exemplo, favorecer timeouts longos ou curtos)?; e (iii) quão eficientes são as estratégias no geral (ou

Estratégia	Propostas	Natureza do Timeout	Algoritmo de cache	Conhecimento sobre o estado do fluxo	Descrição
ST	—	Estático / por inatividade	Aleatório	Baixo	Determinar timeout adequado e aplicá-lo para todas as regras
PEV	[Zarek 2012, Kannan and Banerjee 2014]	Estático / por inatividade	LRU	Alto	Usar timeout longo e periodicamente eliminar regras inativas
AI	[Vishnoi et al. 2014]	Dinâmico / por inatividade	Aleatório	Baixo	Aumentar o timeout quando a regra for reinstalada
AHT	[Zhang et al. 2014]	Dinâmico / rígido	—	Alto	Adaptar o valor do timeout baseado em características de tráfego, tamanho da tabela e desempenho esperado da rede
OPT	—	Dinâmico / rígido	Pacote mais distante no futuro	Alto	Possui, como premissa, conhecimento sobre o tempo de criação, finalização e chegada de pacotes para todos os fluxos

Tabela 2. Sumário de estratégias para gerenciamento temporal de regras.

seja, quão próximas estão do caso ótimo)?

O restante da seção está organizado como segue. Na Seção 4.1, a metodologia de avaliação (incluindo os fatores, as métricas, a carga de trabalho e o ambiente de testes usados nos experimentos) é apresentada. Em seguida, a Seção 4.2 descreve o processo de configuração dos parâmetros de cada estratégia. Por fim, na Seção 4.3 os resultados obtidos são discutidos.

4.1. Metodologia

Ambiente de testes. Os experimentos foram realizados usando-se Mininet HiFi (versão 2.1.0), POX (versão 0.1.0) e Open vSwitch (versão 1.9.0, com suporte a OpenFlow 1.0) para emular uma rede real. A rede emulada foi criada em uma máquina Intel i7 com 4 núcleos de 3.1 GHz e 16 GB de RAM. Nesse cenário, as estratégias oriundas da literatura (ST, PEV, AI e AHT) foram implementadas como aplicações para o controlador POX. O cenário ótimo (OPT), por sua vez, foi avaliado à parte através de simulações. Para facilitar a reprodução dos resultados, os códigos-fonte tanto das aplicações quanto do simulador utilizado encontram-se disponíveis online[1].

A topologia usada é constituída por dois hospedeiros (um cliente e um servidor) conectados por um único switch. Tal configuração permite isolar o comportamento das estratégias e ilustra de modo realista um cenário de gargalo de rede pelo qual grande número de fluxos necessita passar. Para desencadear a geração de regras de encaminhamento no controlador e o preenchimento da tabela de fluxos, foram criadas múltiplas conexões TCP, cada qual associada a um processo diferente nos hospedeiros. Nesse sentido, não houve instalação proativa de regras.

Carga de Trabalho. Para cada experimento, o cliente comunica-se com o servidor de acordo com dois parâmetros: duração do fluxo e intervalo de tempo entre rajadas de pacotes. A duração dos fluxos foi modelada como um processo log-normal com média $\mu = 4$s e desvio padrão $\sigma = 1$s. O intervalo de tempo entre rajadas, por sua vez, segue um processo de Poisson com média $\lambda = 10$s. Tal configuração baseia-se em medições realizadas por trabalhos presentes na literatura [Benson et al. 2010, Karagiannis et al. 2004]. Em particular, a mesma reproduz um cenário realista onde a maioria dos fluxos tem curta duração e transmite pequenas quantidades de dados, enquanto alguns poucos fluxos (de duração mais longa) são responsáveis por boa parte do tráfego.

[1] https://github.com/mcnevesinf/SDN-timebased-rulemanagement

A duração mínima dos fluxos e o menor intervalo de tempo entre rajadas foram definidos como 1s. Valores menores não alterariam os resultados obtidos, visto que esta é a granularidade máxima de tempo empregada em switches OpenFlow [ONF 2015]. O número de fluxos ativos na rede foi fixado em 300 (que se traduz em 600 regras considerando ambas as direções de uma conexão TCP), seguindo a variação no tamanho da tabela conforme descrito a seguir. Por fim, cada experimento teve duração de 10 minutos.

Tamanho da tabela. A capacidade da tabela de fluxos (em número de entradas) foi variada no intervalo [400,800]. De acordo com [Kuzniar et al. 2015, He et al. 2015], essa faixa de valores está em linha com dispositivos comerciais (de baixa capacidade). Esse intervalo foi escolhido, também, de maneira a evitar interferência decorrente de sobrecargas no ambiente de avaliação experimental. Observe que mais importante do que o número absoluto de entradas na tabela é a relação entre o seu tamanho e o número de fluxos. Variando o tamanho da tabela no intervalo escolhido, mediante uma carga fixa, é possível isolar o impacto de cada propriedade (timeout e algoritmo de cache) no desempenho das estratégias para diferentes estados operacionais da rede.

Métricas. O desempenho de cada estratégia foi avaliado segundo duas métricas, conforme a literatura [Zarek 2012]: número de intervenções do controlador e utilização da tabela de fluxos. Por intervenções do controlador, considerou-se tanto as operações de inserção de regras na tabela de fluxos (*flowmods*) quanto as ações de encaminhamento direto de pacotes (*packet outs*). O segundo caso é específico da estratégia AHT. A eliminação de regras (necessária na aplicação dos algoritmos de cache) foi desconsiderada, pois não influencia no desempenho dos fluxos [Bifulco and Matsiuk 2015]. O número de intervenções do controlador por fluxo é calculado pela adição entre o número de inserções (N_I) e o número de encaminhamentos diretos de pacotes (N_E), dividido pelo conjunto de todos os fluxos (F): $\frac{N_I+N_E}{|F|}$. Uma vez que cada intervenção aumenta substancialmente a latência do fluxo associado [Shirali-Shahreza and Ganjali 2015], valores menores são melhores.

A utilização da tabela de fluxos do switch, por sua vez, representa a porcentagem de entradas ocupadas (E_A) sobre o número total de entradas (E_T): $\frac{E_A}{E_T} * 100\%$. Para essa métrica, os valores devem ser os menores possíveis por duas razões: (i) a latência das operações de controle da tabela (i.e., inserção, remoção e modificação de entradas) aumenta de acordo com o número de regras instaladas [Kuzniar et al. 2015, He et al. 2015]; e (ii) quanto maior a utilização da tabela, maior a probabilidade que a mesma fique cheia (principalmente durante rajadas de fluxos) e, portanto, maiores as chances de intervenções do controlador serem necessárias [Guo et al. 2015].

4.2. Configurações Básicas

Para a realização de uma comparação justa, valores apropriados foram estabelecidos empiricamente para os parâmetros de cada estratégia. Esses valores foram definidos através de medições, considerando-se todas as combinações possíveis (avaliação fatorial completa) dos parâmetros listados na Tabela 3. A metodologia utilizada nesses experimentos preliminares também segue àquela descrita na seção anterior. Além disso, verificou-se uma tendência de piora nos resultados para a maioria dos parâmetros inteiros à medida que seus valores se aproximam dos limites testados, indicando que as faixas consideradas são representativas.

Parâmetros	Valores testados
Timeout Estático (ST)	
t_{ST} (em segundos)	5, 10, **20**, 30, 60
Remoção Antecipada Probabilística (PEV)	
t_{PEV} (em segundos)	5, 10, **20**, 30, 60
DI (em segundos)	5, 20, **60**
$P(i)$	CDF da duração dos fluxos **CDF do intervalo de tempo de chegada de pacotes**
Incremento Adaptativo (AI)	
t_{AI}^{min} (em segundos)	1, 2, 5, 10, **20**, 30
t_{AI}^{max} (em segundos)	30, **60**, 120
$f(n\_reinstall)$	Linear: $t_{AI}^{min} \times n\_reinstall$ **Exponencial:** $t_{AI}^{min} \times 2^{n\_reinstall}$
Timeout Rígido Adaptativo (AHT)	
λ μ	De acordo com a carga de trabalho
$C = c_1, c_2, ..., c_n$	Não foram consideradas restrições de desempenho da rede
$t_{AHT}(\lambda, \mu, R, [C])$ (em segundos)	10, 20, **30 (400-650)**, **40 (700)**, **50 (750)**, **60 (800)** 70, 80, 90, 100

Tabela 3. Resultado da análise de sensibilidade dos parâmetros.

A Tabela 3 também destaca as melhores configurações alcançadas para cada estratégia (valores em negrito). Note que a estratégia AHT apresentou configurações distintas para diferentes capacidades da tabela de fluxos (indicadas entre parênteses). Esse comportamento ocorre devido à estratégia considerar o tamanho da tabela no cálculo de duração das regras.

4.3. Resultados

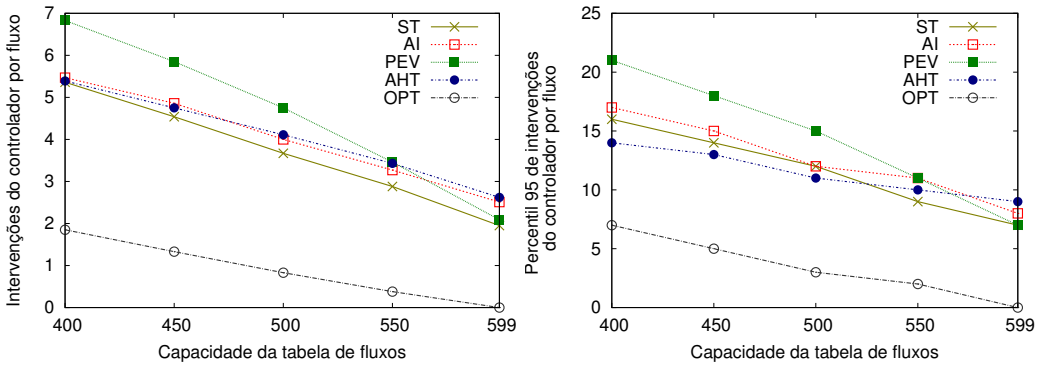
Os resultados estão organizados de acordo com três cenários distintos de operação da rede: escassez (Seção 4.3.1) e superprovisionamento (Seção 4.3.2) de recursos, ambos assumindo um tráfego estável, e situações de alta variabilidade nas características do tráfego (Seção 4.3.3). Um sumário com as principais observações pode ser encontrado na Seção 6.

4.3.1. Dispositivos de recursos limitados

Consideram-se dispositivos de recursos limitados os elementos de rede (switches e roteadores) cuja capacidade das tabelas de fluxos é menor que a demanda de regras durante a maior parte do período de operação da infraestrutura. Na prática, esses dispositivos normalmente envolvem os switches OpenFlow implementados em hardware com o uso de memória TCAM (*Ternary Content Addressable Memory*) e podem ser encontrados principalmente no núcleo das redes [Akyildiz et al. 2014].

Uma vez que a demanda (em número de regras) é maior que a capacidade das tabelas de fluxos, o componente dominante no desempenho das estratégias em cenários de recursos limitados é o seu algoritmo de cache. A Figura 1(a) mostra o número médio de intervenções do controlador por fluxo. Note que estratégias com diferentes algoritmos de cache apresentam desempenhos significativamente diferentes. Surpreendentemente, o algoritmo LRU teve um desempenho pior que o caso aleatório, ao contrário do que é indicado pela literatura [Zarek 2012].

A principal explicação é que o algoritmo LRU funciona bem quando o objetivo é diferenciar entradas ativas de entradas permanentemente inativas (ou com grande período de inatividade). Contudo, ele não tem um desempenho satisfatório quando todos os elementos do sistema (no caso, regras) encontram-se altamente ativos, e provavelmente serão



(a) Média de intervenções do controlador em um fluxo.

(b) Percentil 95 de intervenções do controlador em um fluxo.

Figura 1. Dispositivos de recursos limitados

utilizados ao menos mais uma vez. Nesse caso, espera-se que esquemas diretamente baseados no intervalo de tempo entre pacotes possam apresentar resultados melhores (vide estratégia OPT).

Conforme demonstrado pela Figura 1(a), à medida que a capacidade do dispositivo se aproxima da demanda (valores próximos de 600), mesmo estratégias com algoritmos de cache idênticos (por exemplo, ST e AI) tendem a apresentar desempenhos distintos. Esse comportamento evidencia o aumento da influência do mecanismo de timeout adotado pelas estratégias no seu desempenho. Note que todas as estratégias tiveram um desempenho pelo menos 2x pior que a estratégia ótima (OPT) para todos os cenários de baixa capacidade, mostrando que ainda há espaço para o desenvolvimento de algoritmos de cache mais eficientes.

Ademais, o número médio de intervenções por fluxo foi consideravelmente alto, sendo que grande parte dos fluxos (mais de 90% para uma tabela com capacidade de 400 entradas) teve a necessidade de pelo menos uma intervenção do controlador. O número de intervenções também variou bastante entre os fluxos (como mostrado na Figura 1(b)). De acordo com os resultados, pelo menos 5% dos fluxos apresentaram mais que o dobro da média de intervenções em todos os casos testados. Esse comportamento evidencia a necessidade da implantação de algum mecanismo de justiça entre os fluxos, principalmente em cenários onde os mesmos possuam algum tipo de *deadline* associado [Zhang et al. 2015].

4.3.2. Dispositivos de grande capacidade

Em contraste aos dispositivos de recursos limitados, os dispositivos de grande capacidade representam os elementos de rede cuja capacidade da tabela de fluxos é igual ou superior à demanda de regras durante grande parte do período de operação da rede. Na prática, esse grupo de dispositivos normalmente compreende os switches OpenFlow implementados em hardware com o uso de memória RAM e os software switches (por exemplo, Open vSwitch [Pfaff et al. 2015]). Tais dispositivos tendem a possuir taxas menores de processamento de pacotes que os dispositivos de recursos limitados e, por essa razão, geralmente são utilizados na borda das redes (por exemplo, junto aos hipervisores em redes

de *data center*) [Akyildiz et al. 2014].

Nos casos de grande capacidade, o componente determinante no desempenho das estratégias tende a ser o seu mecanismo de timeout, uma vez que a tabela tem menos chances de ficar cheia e, por consequência, os algoritmos de cache são pouco acionados. A exceção são os cenários de timeout excessivamente longos, que tendem a manter regras inativas nas tabelas.

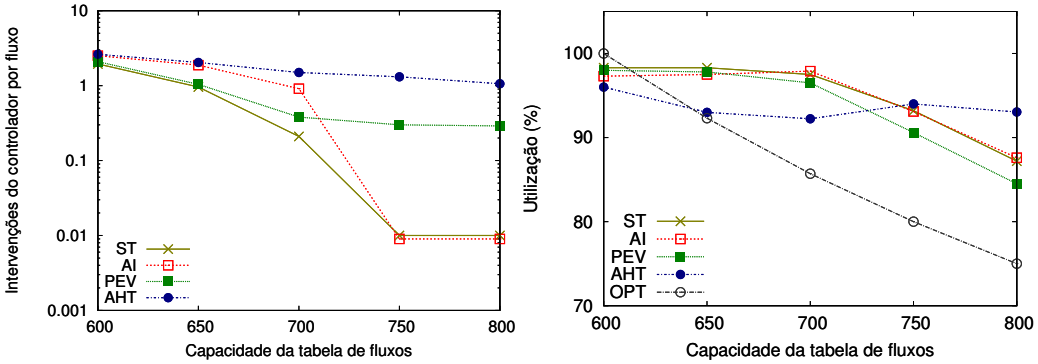
A Figura 2(a) mostra o número médio de intervenções do controlador por fluxo para cenários com dispositivos de grande capacidade, estando o eixo y representado em escala logarítmica. Note que a estratégia OPT está ausente porque não há necessidade de intervenções do controlador para o caso ótimo, uma vez que é possível manter cada regra instalada durante a duração exata do respectivo fluxo. Com relação às demais estratégias, a maioria apresenta uma quantidade baixa de intervenções (menos de 1 por fluxo na média), com um percentual significativo dos fluxos não necessitando de intervenção alguma. A principal conclusão é que, com uma parametrização adequada, é possível atingir um alto grau de eficiência no gerenciamento temporal das regras, mesmo quando estratégias simples (como o uso isolado de timeout estático) são consideradas (vide ST). Afinal, existe sempre um tamanho de tabela suficientemente grande para comportar todas as regras e evitar a necessidade de uma nova intervenção do controlador.

A Figura 2(b), por sua vez, mostra a utilização média da tabela de fluxos para cada experimento realizado. A maioria das estratégias tende a utilizar entre 10 e 15% a mais os recursos em comparação ao caso ótimo (OPT). De maneira geral, a utilização relaciona-se inversamente e de forma não-linear com o número de intervenções do controlador (ou seja, uma pequena redução no uso da tabela pode implicar em um aumento significativo da necessidade de intervenções). Considerando as estratégias analisadas, a tentativa de retirada antecipada de uma regra tende a amplificar mais de 10x (vide PEV) a necessidade de intervenções do controlador. O mesmo efeito se aplica ao uso de timeouts excessivamente curtos. Note que a estratégia OPT usa inicialmente 100% da capacidade da tabela, uma vez que ela não tenta remover regras de fluxos que ainda podem estar ativos havendo espaço suficiente.

Considerando que a latência de comunicação com o controlador é tipicamente uma a duas ordens de grandeza maior que a latência de uma operação de controle sobre a tabela (por exemplo, inserção ou modificação de uma regra) [Bifulco and Matsiuk 2015], estratégias que priorizam a redução no número de intervenções em detrimento da utilização (por exemplo, ST e AI) são preferíveis. Por fim, estratégias baseadas em mecanismos rígidos de timeout tendem a apresentar tanto índices de utilização quanto de intervenções elevados, mesmo considerando características da carga na sua definição (vide AHT). Provavelmente esse efeito é decorrente da granularidade (baixa) desse tipo de timeout, que requer escalas maiores de valores.

4.3.3. Cenários de parametrização imprecisa

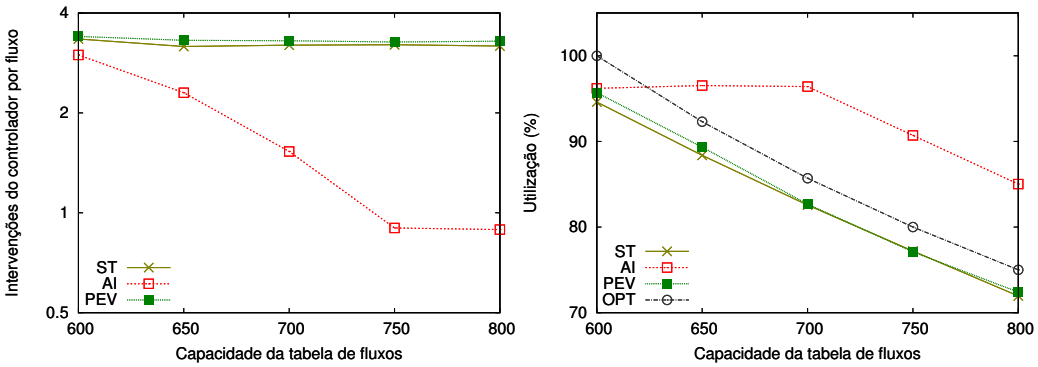
Diversos estudos apontam variabilidades nas características do tráfego das redes quando longos períodos (isto é, ordens de grandeza de segundos ou maiores) são considerados [Cordero and Bonaventure 2014, Almeida et al. 2015, Duarte et al. 2014]. Nesse sentido,



(a) Média de intervenções do controlador em um fluxo.

(b) Utilização média da tabela de fluxos.

Figura 2. Dispositivos de grande capacidade.



(a) Média de intervenções do controlador em um fluxo.

(b) Utilização média da tabela de fluxos.

Figura 3. Cenários de parametrização imprecisa.

imprecisões podem ocorrer com relação à parametrização das estratégias. Nesta seção, os principais resultados obtidos para cenários imprecisos são discutidos.

De forma geral, os parâmetros podem estar subestimados ou superestimados, resultando na ocorrência de regras com durações excessivamente curtas ou longas, respectivamente. Nesse contexto, a Figura 3 mostra o número de intervenções do controlador e a utilização da tabela de fluxos quando os parâmetros estão subestimados. Para esses cenários, adotou-se a configuração de parâmetros que levou aos resultados mais próximos daqueles obtidos na Seção 4.2 para o melhor caso. Em outras palavras, $t_{ST} = t_{PEV} = t_{AI}^{min} = 10s$. Os demais parâmetros não foram alterados. Finalmente, a estratégia AHT não foi considerada, uma vez que ela leva em conta a carga da rede na determinação da duração das regras.

Pode-se perceber que pequenas mudanças na qualidade da parametrização das estratégias (ou, de forma equivalente, nas características do tráfego) podem ocasionar grandes variações de desempenho (Figuras 2 e 3). Nesse contexto, estratégias envolvendo o uso de timeouts dinâmicos tendem a apresentar melhores resultados, uma vez que é possível adaptá-las à carga da rede de modo mais eficiente. O processo de adaptação, no entanto, também tem um custo não negligível, percebido quando comparamos as Figuras 2(a) e 3(a) (vide estratégia AI) e os índices de utilização da tabela de fluxos (Figura

3(b)). Note que o algoritmo que minimiza o número de intervenções (OPT) pode levar, por outro lado, a níveis de ocupação da tabela maiores. Ademais, quando parâmetros superestimados são considerados, o mesmo comportamento geral se repete.

5. Trabalhos Relacionados

Além do grupo de estratégias estudado neste artigo, a literatura sobre gerenciamento de regras em redes SDN compreende propostas que buscam reduzir a quantidade total de regras necessárias para a implantação de uma determinada política. Nesse sentido, [Voellmy et al. 2013] e [Schlesinger et al. 2014] tentam otimizar o processo de transformação de políticas de alto nível nas suas respectivas regras através do uso de *frameworks* (isto é, linguagens e compiladores) específicos para a expressão dessas políticas. [Kang et al. 2013] e [Moshref et al. 2013], por sua vez, atuam posicionando estrategicamente as regras na rede de forma a permitir que cada regra possa ser aplicada ao maior número possível de fluxos, bem como a minimizar a sobrecarga de tráfego gerada com prováveis desvios nos caminhos dos pacotes. Tais propostas são ortogonais às avaliadas neste trabalho, podendo ambos os grupos serem implantados de forma conjunta e independente em um ambiente de rede SDN.

6. Conclusão

Neste artigo, analisamos o espaço de projeto e comparamos o desempenho de diferentes estratégias de domínio temporal para o gerenciamento de regras em redes SDN. Este é o primeiro trabalho a fazer um estudo sistemático e abrangente da área. As observações-chave obtidas foram as seguintes:

- Conforme esperado, quando há um superprovisionamento de recursos na rede, a maioria das estratégias apresenta uma quantidade baixa de intervenções do controlador (menor que uma por fluxo na média), com um percentual significativo dos fluxos não necessitando de intervenção alguma. Em contrapartida, elas tendem a ter um nível de utilização entre 10 e 15% maior de recursos em comparação com a estratégia OPT.
- A utilização da tabela relaciona-se de forma inversa e *não-linear* com o número de intervenções do controlador (que cresce rapidamente). Dessa forma, regras de duração longa são preferíveis em cenários de incerteza.
- Pequenas mudanças nas características do tráfego podem afetar consideravelmente o desempenho das estratégias. Assim, estratégias envolvendo o uso de timeouts dinâmicos tendem a apresentar melhores resultados, pois podem se adaptar às variações da carga. Contudo, mesmo o processo de adaptação possui um custo não negligível.
- Em cenários de escassez de recursos, o desempenho das estratégias varia significativamente de acordo com o algoritmo de cache utilizado. No geral, todas as estratégias foram pelo menos 2x piores que o caso ótimo com relação ao número de intervenções do controlador. Além disso, houve uma ampla variação do número de intervenções entre os fluxos.

Acreditamos que tais observações poderão qualificar o desenvolvimento de novos estudos e alimentar avanços científicos na área, fundamental para o uso eficiente dos recursos de encaminhamento da rede. Como trabalhos futuros, pretende-se estender

a avaliação realizada para considerar outros fatores, tais como múltiplas tabelas de fluxos, diferentes arquiteturas de cache e cenários mais complexos de rede, que envolvam agregação de regras (regras com wildcards), topologias maiores e dispositivos de encaminhamento heterogêneos. Também pretende-se investigar mecanismos para melhorar o desempenho das estratégias estudadas, potencialmente adicionando-se formas de oferecer garantias de desempenho aos fluxos (por exemplo, tempo máximo de conclusão).

Agradecimentos. Essa pesquisa é financiada em parte pelos projetos Phoenix (MCTI/CNPq 14/2014) e ProSeG (MCTI/CNPq/CT-ENERG 33/2013).

Referências

Akyildiz, I. F., Lee, A., Wang, P., Luo, M., and Chou, W. (2014). A roadmap for traffic engineering in sdn-openflow networks. *Computer Networks*, 71:1 – 30.

Almeida, W., Santos, B., Vieira, A., Cunha, I., and Almeida, J. (2015). Caracterização da transmissão de um grande evento esportivo. In *33o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2015*.

Benson, T., Akella, A., and Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement - IMC'10*.

Bifulco, R. and Matsiuk, A. (2015). Towards scalable sdn switches: Enabling faster flow table entries installation. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM'15*.

Cordero, J. and Bonaventure, O. (2014). Understanding the topological properties of internet traffic: A view from the edge. In *2014 IFIP Networking Conference*.

Duarte, R., Cunha, I., Almeida, J., and Vieira, A. (2014). Avaliação do impacto de falhas na rede nacional de ensino e pesquisa no tráfego de um campus universitário. In *32o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2014*.

Guo, Z., Xu, Y., Cello, M., Zhang, J., Wang, Z., Liu, M., and Chao, H. J. (2015). Jumpflow: Reducing flow table usage in software-defined networks. *Computer Networks*, 92:300 – 315.

He, K., Khalid, J., Gember-Jacobson, A., Das, S., Prakash, C., Akella, A., Li, L. E., and Thottan, M. (2015). Measuring control plane latency in sdn-enabled switches. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research - SOSR'15*.

Kang, N., Liu, Z., Rexford, J., and Walker, D. (2013). Optimizing the "one big switch"abstraction in software-defined networks. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies - CoNEXT'13*.

Kannan, K. and Banerjee, S. (2014). Flowmaster: Early eviction of dead flow on sdn switches. In *LNCS, Distributed Computing and Networking*, volume 8314, pages 484 – 498.

Karagiannis, T., Molle, M., Faloutsos, M., and Broido, A. (2004). A nonstationary poisson view of internet traffic. In *2004 IEEE Conference on Computer Communications - INFOCOM'04*.

Katta, N., Alipourfard, O., Rexford, J., and Walker, D. (2014). Infinite cacheflow in software-defined networks. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking - HotSDN'14*.

Kuzniar, M., Peresini, P., and Kostic, D. (2015). What you need to know about sdn flow tables. In *LNCS, Passive and Active Measurement*, volume 8995, pages 347 – 359.

Moshref, M., Yu, M., Govindan, R., and Vahdat, A. (2014). Dream: Dynamic resource allocation for software-defined measurement. In *Proceedings of the 2014 ACM Conference on Special Interest Group on Data Communication - SIGCOMM'14*.

Moshref, M., Yu, M., Sharma, A., and Govindan, R. (2013). Scalable rule management for data centers. In *10th USENIX Symposium on Networked Systems Design and Implementation - NSDI'13*.

Nguyen, X.-N., Saucez, D., Barakat, C., and Turletti, T. (2015). Officer: A general optimization framework for openflow rule allocation and endpoint policy enforcement. In *2015 IEEE Conference on Computer Communications - INFOCOM'15*.

ONF (2015). Openflow switch specification. version 1.5.1 (protocol version 0x06). Disponível em: `https://www.opennetworking.org`. Accesso em Novembro, 2015.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., and Casado, M. (2015). The design and implementation of open vswitch. In *12th USENIX Symposium on Networked Systems Design and Implementation - NSDI'15*.

Schlesinger, C., Greenberg, M., and Walker, D. (2014). Concurrent netcore: From policies to pipelines. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming - ICFP'14*.

Shirali-Shahreza, S. and Ganjali, Y. (2015). Rewiflow: Restricted wildcard openflow rules. *SIGCOMM Comput. Commun. Rev.*, 45(5):29–35.

Su, Z., Wang, T., Xia, Y., and Hamdi, M. (2015). Cemon: A cost-effective flow monitoring system in software defined networks. *Computer Networks*, 92:101 – 115.

Vishnoi, A., Poddar, R., Mann, V., and Bhattacharya, S. (2014). Effective switch memory management in openflow networks. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems - DEBS'14*.

Voellmy, A., Wang, J., Yang, Y. R., Ford, B., and Hudak, P. (2013). Maple: Simplifying sdn programming using algorithmic policies. In *Proceedings of the 2013 ACM Conference on Special Interest Group on Data Communication - SIGCOMM'13*.

Zarek, A. (2012). OpenFlow Timeouts Demystified. Master's thesis, University of Toronto, Canada.

Zhang, H., Shi, X., Yin, X., Ren, F., and Wang, Z. (2015). More load, more differentiation — a design principle for deadline-aware congestion control. In *2015 IEEE Conference on Computer Communications - INFOCOM'15*.

Zhang, L., Lin, R., Xu, S., and Wang, S. (2014). AHTM: Achieving efficient flow table utilization in software defined networks. In *2014 IEEE Global Communications Conference - GLOBECOM'14*.

REUNI: Um algoritmo para REduzir tabelas de encaminhamento e UNIformizar a distribuição dos fluxos em redes com topologia hipercubo

Dione S. A. Lima[1], Rafael S. Guimarães[12], Alextian B. Liberato[12]
Eros Spalla[2], Gilmar L. Vassoler[2], Magnos Martinello[1], Rodolfo S. Villaça[1] *

[1] Núcleo de Estudos em Redes Definidas por Software (NERDS)
Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo (UFES) – Vitória/ES

[2]Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo (IFES)
Campus Cachoeiro/ES, Colatina/ES, Aracruz/ES, Serra/ES – Brazil

dione.sousa@gmail.com, {rafaelg,alextian,spalla}@ifes.edu.br
gilmar@ifes.edu.br, magnos@inf.ufes.br, rodolfo.villaca@ufes.br

Abstract. *The hypercube is an interconnection topology of servers in a Data Center. In this topology, packet forwarding are normally based on XOR operation, which is highly efficient when the hypercupe is complete, but doesn't work when there are node or link failures. The use of SDN and OpenFlow can be viewed as an alternative to ensure traffic forwarding in these situations. However, the adoption of forwarding tables in a hypercube has a high cost associated to the huge number of entries in these tables, which increases as an exponential function of the number of nodes. In this context, this paper presents REUNI, an algorithm to reduce the number of entries in the forwarding table of a hypercube. REUNI achieves a logarithmic compression rate, which turns the number of entries equal to the number of dimensions of the hypercube. In addition to the compression rate, the distribution of the number of flows in the links was also evaluated and it was observed that the number of flows in a communication among all nodes are equally distributed between the links in the hypercube.*

Resumo. *Uma possível topologia de interligação dos servidores em um Data Center é o hipercubo. Nele, o encaminhamento de pacotes é normalmente baseado em operações de XOR e é bastante eficiente, porém só funciona na ausência de falhas de nós ou enlaces no hipercubo. O uso de SDN e do protocolo OpenFlow pode ser uma alternativa para garantir o encaminhamento de tráfego nessas situações. Entretanto, a adoção de tabelas de encaminhamento em um hipercubo possui um alto custo, relacionado ao grande número de entradas nessas tabelas, que crescem em escala exponencial. Nesse contexto, este artigo apresenta o REUNI, um algoritmo para redução do número de entradas dessas tabelas com taxa de compactação logarítmica, proporcional ao número*

*Este trabalho tem recebido financiamento do projeto Horizon 2020 da União Européia para pesquisa, desenvolvimento tecnológico e demonstração sob no. 688941 (FUTEBOL), assim como do Ministério Brasileiro da Ciência, Tecnologia e Inovação (MCTI) por meio da RNP e do CTIC. Além disso, gostaríamos de agradecer o financiamento do CNPq sob no. 456143/2014-9 e 449369/20145, e da FAPES sob no. 524/2015.

de dimensões do hipercubo. Além da taxa de compactação, avaliou-se também a distribuição dos fluxos nos enlaces após a compactação e observou-se que após a compactação das tabelas ocorre um balanceamento natural das rotas nos enlaces do hipercubo em uma comunicação de todos para todos.

1. Introdução

Data Centers são bastante utilizados no processamento e armazenamento de grandes volumes de dados. Em função disso, existem diversas arquiteturas para a montagem da sua infraestrutura tais como: Dcell [Guo et al. 2008], BCube [Guo et al. 2009] e Fat-Tree [Al-Fares et al. 2008]. Em particular, nas arquiteturas de *Data Centers* centrados em servidores (*server-centric*), a característica mais importante é que os elementos de rede (*switches*, por exemplo) não estão envolvidos diretamente nas decisões de encaminhamento de tráfego de rede, eles simplesmente fornecem conexões entre os servidores, e estes sim, são os responsáveis por todas as decisões de encaminhamento no interior do *Data Center*. Nesse contexto, equipamentos de rede de baixo custo (COTS, ou *Commodity Off-the-Shelf*) podem ser utilizados na montagem da infraestrutura, até mesmo ligações diretas entre os servidores podem ser estabelecidas nestes *Data Centers*.

Uma possível topologia de interligação dos servidores nas arquiteturas de *Data Center* centrados em servidores é o hipercubo, e neste contexto podemos destacar o TRI-IIAD, proposto em [Vassoler 2015]. Em um hipercubo cada servidor (que também pode ser chamado de um nó da rede do *Data Center*) possui um identificador único de n-bits e n interfaces de rede, que o interliga a outros n servidores, que serão os seus vizinhos no *Data Center*. A restrição na escolha desses vizinhos é tal que cada servidor só poderá ter como vizinhos outros servidores cuja distância de Hamming (D_h) entre seus identificadores é igual a 1. Essa restrição facilita o processo de encaminhamento de tráfego no interior do *Data Center*, onde basta uma simples operação de ou-exclusivo (XOR) para determinação da interface de saída para a qual os pacotes deverão ser encaminhados a fim de alcançarem o seu destino.

Embora o encaminhamento baseado em operações XOR seja bastante eficiente [Dally 1990, Vassoler 2015], o mesmo só funciona em situações onde os hipercubos virtuais estão completos, ou seja, na ausência de falhas de nós ou de enlaces no *Data Center*. Na ocorrência de uma falha como essas o resultado da operação XOR poderá indicar uma interface de saída que leve o pacote a um nó inoperante ou a um enlace indisponível, o que irá ocasionar perda de pacotes e quebra da completa conectividade entre os nós do *Data Center* enquanto a falha estiver presente. Em um trabalho anterior [de Lima et al. 2015] apresentamos uma abordagem baseada em Redes Definidas por Software (SDN, ou *Software Defined Networking*), especificamente no protocolo OpenFlow [McKeown et al. 2008], para garantir o encaminhamento de tráfego em *Data Centers* centrados nos servidores ligados na topologia de hipercubo, mesmo em situações de falha.

Entretanto, a adoção de tabelas de encaminhamento em *Data Centers* com topologia baseada em hipercubo possui um alto custo relacionado ao grande número de entradas nessas tabelas, que tende a crescer exponencialmente em função de seu grau (n). Em um *Data Center* com topologia em hipercubo existe uma enorme multiplicidade de caminhos entre quaisquer pares origem-destino [Saad and Schultz 1989], mas que o uso de encaminhamento sem estado (XOR) traz problemas relacionados com a tolerância à

falhas [de Lima et al. 2015]. Além disso, a adoção do XOR como algoritmo de encaminhamento também provoca um forte desbalanceamento na distribuição de tráfego entre os enlaces disponíveis no *Data Center*. Estes resultados serão apresentados na Seção 4 deste artigo.

Desta forma, baseando-se nos trabalhos propostos em [Vassoler 2015, de Lima et al. 2015], este artigo apresenta o REUNI, um algoritmo para redução do número de entradas nas tabelas de encaminhamento dos nós de um *Data Center* centrado em servidores com topologia hipercubo. A proposta apresentada neste artigo faz uso de máscaras de bit (*bitmaks*), característica disponível no OpenFlow a partir de sua versão 1.1, e de técnicas de compactação de tabelas de encaminhamento baseadas na correspondência pelo prefixo mais longo. O REUNI alcança uma taxa de compactação de escala logarítmica (igual a $log_2(2^n) = n$), independentemente da existência de virtualização de servidores, além de propiciar uma melhor distribuição dos fluxos nos enlaces do hipercubo com a garantia de encaminhamento sempre pelo caminho mais curto (SPF, ou *Shortest Path First*).

Para validar a proposta foram feitos comparativos entre o número de entradas nas tabelas de encaminhamento de *Data Centers* em topologia hipercubo, compatíveis com as propostas de [Vassoler 2015, de Lima et al. 2015], com $3 \leq n \leq 10$, com e sem compactação nas tabelas, com e sem virtualização de servidores. Para comprovar a distribuição de carga desbalanceada entre os enlaces quando se utiliza encaminhamento sem estado baseado em XOR, avaliou-se a distribuição do número de fluxos em cada enlace do hipercubo em cenários de comunicação de todos para todos os servidores. Em seguida, a avaliação foi rodada novamente considerando-se as tabelas compactadas com o algoritmo proposto. Por fim, avaliou-se o tempo gasto para compactação de todas as tabelas para todos os servidores.

A principal contribuição deste artigo é o algoritmo de compactação das tabelas de encaminhamento (REUNI), que garante a redução do número de entradas em escala logarítmica com garantia de entrega sempre pelo menor caminho. Na Seção 2 são apresentados importantes trabalhos relacionados e suas contribuições. O algoritmo proposto e detalhes de sua implementação estão apresentados na Seção 3. A Seção 4 apresenta os resultados das avaliações feitas no protótipo desenvolvido com hipercubos com até 1024 nós. Por fim, na Seção 5 são tecidas as conclusões e trabalhos futuros para aprimoramento da proposta.

2. Trabalhos Relacionados

O maior problema relacionado com o uso da topologia hipercubo em *Data Centers* centrados em servidores é a sua incapacidade no tratamento de falhas usando a operação XOR como método de encaminhamento, apesar de sua baixíssima latência [Vencioneck et al. 2014]. A alternativa de uso de tabelas de encaminhamento garante a conectividade nesses casos, porém tem um alto custo associado ao seu crescimento exponencial em função do número de nós do hipercubo.

TRIIIAD [Vassoler 2015] é um trabalho relacionado do grupo NERDS que, dentre outras coisas, demonstra a viabilidade e as vantagens do uso de *Data Centers* programáveis com topologia hipercubo. Na arquitetura TRIIIAD os nós do hipercubo são servidores de grande porte que hospedam máquinas virtuais (VMs) e suas aplicações. Todo o controle de migrações de VMs e orquestração dos fluxos rede é baseado em

controladores SDN em um conceito denominado A-SDN *(Augmented SDN)*, inclusive permitindo a troca do método de encaminhamento de cada nó "*on the fly*" usando o Flex-Forward [Vencioneck et al. 2014]. Toda essa flexibilidade permite que *Data Centers* com poucos nós físicos possam aproveitar-se das vantagens da topologia hipercubo, que são: baixíssima latência de encaminhamento, alto grau de redundância nos caminhos entre os servidores. O REUNI contribui nesse contexto reduzindo, em escala logarítmica, o tamanho das tabelas de encaminhamento dos nós OpenFlow do *Data Center* TRIIIAD, viabilizando sua implementação em ambientes reais.

O BCube [Guo et al. 2009] e o DCell [Guo et al. 2008] são propostas de *Data Center* centrados em servidores modulares mais citada na literatura, cuja topologia é bastante similar ao hipercubo. Uma vez que a topologia não é um hipercubo original, a ocorrência de falhas não ocasiona interrupção de conexão entre os servidores, porém aumenta-se o tamanho médio dos caminhos entre eles e reduz-se a quantidade de opções de caminhos entre os servidores.

Em [Luiz Fernando T. de Farias 2014] é apresentado um algoritmo para compactação de tabelas de roteamento sob a ótica da interface de saída em roteadores IP. Embora também usem a estratégia do prefixo mais longo para agrupar rotas, essa agregação ocorre nos endereços IP. Em [Braun and Menth 2014] é apresentado um trabalho com proposta similar ao de [Luiz Fernando T. de Farias 2014], também com aplicação em roteadores BGP e com o objetivo de reduzir o número de entradas BGP nos roteadores das redes de acesso. Ambos se diferenciam por considerarem a agregação no nível de rede (IP), enquanto que o REUNI aplica a redução em endereços MAC, usados pela arquitetura TRIIIAD para endereçar nós e VMs, conforme descrição na Seção 3.

Considerando os trabalhos relacionados previamente citados, o REUNI avança o estado da arte no contexto dos *Data Centers* centrados em servidores ao viabilizar o uso da topologia hipercubo em *Data Centers* de pequeno e médio porte em redes reais. Essa viabilização dar-se-á pelo alto grau de compactação das tabelas de encaminhamento dos nós e por, naturalmente, colaborar para uma melhor distribuição dos fluxos nessas redes.

3. Implementação

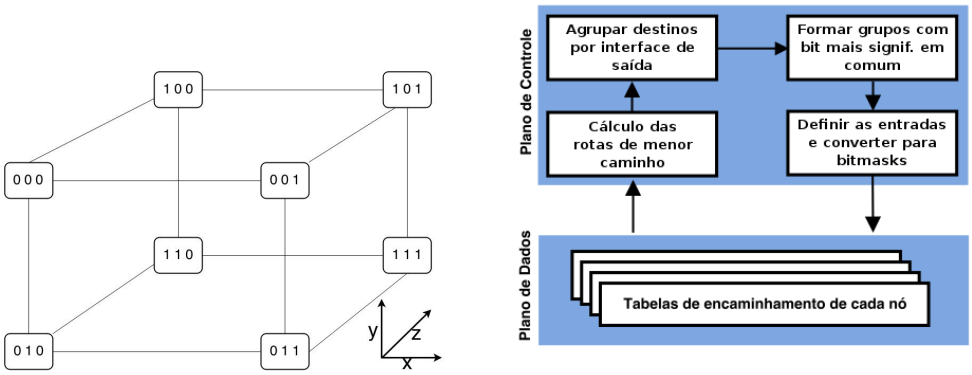
A solução proposta neste artigo compreende os dois aspectos fundamentais das SDNs: plano de controle e plano de dados. No plano de controle serão descritas as características da aplicação no controlador, abordando detalhadamente o REUNI. Dentre as ações realizadas por essa aplicação, destacam-se:

1. Representação topológica do hipercubo;
2. Execução do cálculo das rotas identificando todos os possíveis menores caminhos através do algoritmo (*Shortest Path First*);
3. Implementação do REUNI;
4. Instalação das regras OpenFlow utilizando *bitmasks* no plano de dados.

No plano de dados serão tratadas as questões referentes a instalação das regras de encaminhamento, utilizando-se dos *bitmasks*, característica implementada no OpenFlow 1.3 que permite a utilização de regras curinga (*).

3.1. Plano de Controle

No Plano de Controle a aplicação faz uso de uma representação topológica do hipercubo por meio de uma estrutura de dados do tipo grafo. Para tal, foi utilizada a biblioteca *NetworkX*[1]. A Fig. 1(a) representa um hipercubo grau 3, que será usado para exemplificar o funcionamento do REUNI, onde o bit menos significativo representa o eixo *x*, o segundo bit representa o eixo *y* e o bit mais significativo representa o eixo *z*. Como podemos observar todos os nós são identificados obedecendo o critério da Distância de Hamming (D_h) igual a 1 entre quaisquer pares de nós vizinhos.



(a) (b)

Figura 1. Visão geral: (a) Hipercubo com três dimensões e (b) Construção da aplicação.

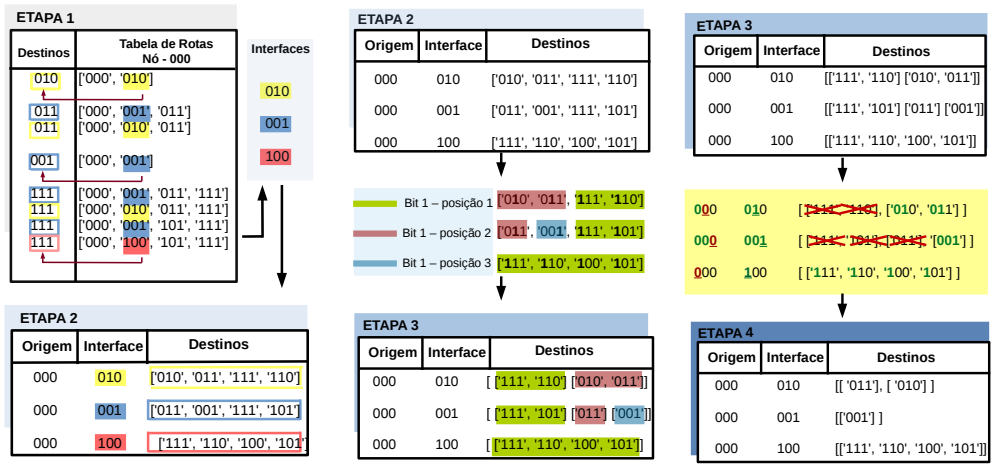
A aplicação no plano de controle será responsável pelo cálculo das rota entre quaisquer pares de origem e destino, identificando, inicialmente, todos os possíveis menores caminhos entre eles. Em seguida as tabelas de encaminhamento de cada nó do hipercubo serão submetidas ao REUNI para compactação. A partir das tabelas compactadas são definidas as novas regras de encaminhamento que serão instaladas em cada nó do hipercubo de acordo com as características e restrições do protocolo OpenFlow. A Figura 1(b) mostra um resumo desse processo, onde o resultado de uma etapa torna-se a entrada para a etapa seguinte. Cada etapa será detalhada a seguir.

A Figura 2 exemplifica todo o processo a partir do nó de origem 000. Na Etapa 1 são geradas as tabelas de encaminhamento de cada nó do hipercubo com todos os caminhos para todos os possíveis destinos do hipercubo. De acordo com a Fig. 2(a), tem-se na primeira coluna da tabela de rotas do nó 000 todos os possíveis destinos no hipercubo a partir desta origem. Por uma questão de limitação de espaço, a tabela da Etapa 1 foi reduzida. Para cada destino, na segunda coluna, são mostradas todas as menores rotas, representadas pelos nós que compõem esse caminho. Por exemplo, a partir nó 000 e com destino ao nó 010 temos a rota $[000, 010]$, ou seja, o destino está diretamente ligado a origem 000 e consequentemente ambos são vizinhos no hipercubo (observe que a D_H entre os identificadores de origem e destino é igual a 1). A partir do mesmo nó 000, porém com destino a 011, por exemplo, observa-se dois caminhos de mesmo custo, que são $[000, 001, 011]$ e $[000, 010, 011]$ (observe que ambos destinos possuem $D_h = 2$ com

[1]http://networkx.github.io

a origem). Isso mostra que esses nós podem se comunicar usando tanto o vizinho 001, quanto o vizinho 010 como próximo salto. A escolha do vizinho para qual o pacote deverá ser encaminhado representa uma interface de saída no hipercubo da Figura 1(a). É usado o termo interface de saída para representar o próximo salto a partir de um nó. A geração desta tabela com todas as possíveis rotas entre origens e destinos corresponde à Etapa 1 do REUNI.

O objetivo da Etapa 2 é: para cada nó, agrupar todos os seus possíveis destinos de acordo com a interface de saída utilizada para alcançá-los, ou seja, agrupa-se os destinos de acordo com o vizinho para o qual o pacote será encaminhado. Para o nó 000, cujas "interfaces" de saída são 010, 001 e 100, são verificados quais os destinos que utilizarão cada um desses vizinhos no encaminhamento dos pacotes a partir do nó 000. Podemos observar na tabela "ETAPA 1" da Fig. 2(a) a identificação das interfaces realçadas na coluna do meio e os destinos sendo associados às mesmas. Em seguida, para cada interface, o REUNI vai agrupar todos os destinos que serão alcançados através de uma mesma interface. Como resultado para o nó 000, temos todas as suas interfaces de saída com suas respectivas listas de destinos. É importante ressaltar aqui que esta é uma grande vantagem da topologia hipercubo: a multiplicidade de caminhos de menor custo entre quaisquer pares de nós da rede.



(a) REUNI - Etapas 1 e 2 (b) REUNI - Etapa 3 (c) REUNI - Etapa 4

Figura 2. REUNI - Etapas 1 a 4

Na Etapa 3, uma lista de destinos é associada a cada interface de saída. Para cada lista é feita uma verificação da posição do bit mais significativo com valor 1 de cada endereço de destino na lista, como pode ser observado na Figura 2(b). Os endereços são agrupados de acordo com essa posição. Por exemplo, observando a lista de destinos pertencentes a interface 010$[[010, 011, 111, 110]]$, o REUNI verifica primeiramente a posição mais significativa do bit 1 de cada endereço da lista e ao encontrar endereços com as mesmas características, adiciona-os em um lista específica que será composta pelos endereços $[111, 110]$. A lista seguinte, $[010, 011]$, é composta com os endereços onde o primeiro bit 1 ocupa a segunda posição mais significativa, e assim sucessivamente, até atingir a quantidade de bits que compõem o endereço. A Função `terceiraEtapa` no Algoritmo 1 representa esse processo. É possível verificar que no exemplo escolhido (nó 000, interface 010) o REUNI não apresenta uma lista cujos bits ocupam a terceira posição

mais significativa, pois a lista de destinos da interface 010 não contém endereços com essa característica.

Algoritmo 1: Terceira e Quarta Etapas

```
Função terceiraEtapa (L_D)
início
    Entrada: L_D ← segundaEtapa()
    Saída: Lista Final com listas de destinos agregados pelo bit mais significativo
    L_1, L_2, ..., L_N
    para cada D_L ∈ L_D faça
        se Operação de bit ∧ D_l na posição N == VERDADEIRO então
            adiciona D_L à lista correspondente da posição L_N
        fim se
    fim para cada
    retorna L_1, L_2, ..., L_N
fim
Função quartaEtapa (L_D)
início
    Entrada: L_D ← terceiraEtapa()
    Saída: Lista Final com lista destinos por interface
    Li_1, Li_2, ..., Li_N
    para cada D_L ∈ L_D faça
        se D_L com maior número de endereços então
            LMaiorTamanho ← D_L
            adiciona LMaiorTamanho a lista de destinos por interface Li_N
        senão
            LAdicional ← (D_L − LMaiorTamanho)
            LFinal ← bit mais significativo da interface Li_N ∧ LAdicional
            adiciona LFinal a lista de destinos por interface Li_N
        fim se
    fim para cada
    retorna Li_1, Li_2, ..., Li_N
fim
```

Na Etapa 4, o REUNI faz uma verificação nas listas de todas as interfaces do seu respectivo nó, no caso do nosso exemplo, do nó 000, para identificar os endereços de destinos que aparecem em listas de mais de uma interface. É possível notar na tabela "ETAPA 1" da Fig. 2(a) que alguns destinos podem ser alcançados por meio de várias interfaces de saída por rotas distintas, ou seja, através de diferentes interfaces. Com isso, torna-se necessário escolher por qual interface aquele destino será alcançado, ou seja, em qual lista o mesmo deverá permanecer. Esse é o objetivo da Etapa 4, representado na Função `quartaEtapa` do Algoritmo 1.

Primeiramente o REUNI verifica qual a interface contem a lista com maior quantidade de endereços de destinos com o bit mais significativo, com valor 1, na mesma posição. No nosso exemplo, a interface 100 contém uma lista com essa característica, onde todos os seus destinos apresentam o valor 1 no bit mais significativo. Esses destinos formam uma nova lista (`LMaiorTamanho` no algoritmo). Em seguida, as listas de destino das outras interfaces são verificadas a fim de eliminar os endereços de destino em comum com essa nova lista, conforme é possíveis visualizar na Figura 2(c).

Por último é feita uma comparação bit a bit entre o endereço do nó de origem, no caso o 000 com o endereço de cada interface. Essa comparação resulta na composição de um prefixo que corresponde aos bits em comum entre o endereço do nó de origem e o endereço da interface mais o primeiro bit diferente do endereço da interface. Por exemplo, comparando o endereço de origem 000, da esquerda para a direita, com o endereço da interface 010, percebe-se que tanto o primeiro bit da origem quanto o primeiro bit do endereço da interface são iguais, então o mesmo será o primeiro bit utilizado para compor

o prefixo. Verifíca-se agora o segundo bit dos dois endereços, onde os mesmos são diferentes. Dessa forma o prefixo será composto pelo bit em comum entre os dois endereços mais o bit diferente do endereço da interface, resultando no prefixo 01. Na comparação, ao encontrar o primeiro bit diferente no endereço da interface, o mesmo será o último bit a compor o prefixo e indicará o fim da comparação dos bits dos endereços em questão. Com isso, todos os endereços pertencentes a lista da interface 010 que iniciam com esse prefixo permanecerão na lista. Os outros endereços serão excluídos. Dessa forma o REUNI provê a tabela compactada para cada nó do hipercubo.

3.2. Plano de Dados

Como pôde ser observado na Figura 1(a), cada nó no hipercubo possui um endereço binário que representa seu identificador e localizador no hipercubo. Na prática, na arquitetura TRIIIAD o endereço de enlace (*MAC Address*) é usado para identificação e localização de cada nó. Como cada nó representa servidores físicos que abrigam máquinas virtuais, uma parte do endereço MAC é utilizado para sua identificação e localização do nó no hipercubo (servidor físico) e outra para a identificação das VMs no hipercubo, conforme visualizado na Fig. 3. Os 32 bits mais significativos são utilizados para caracterizar o servidor físico e outros 16 bits identificam as máquinas virtuais hospedadas neste servidor.

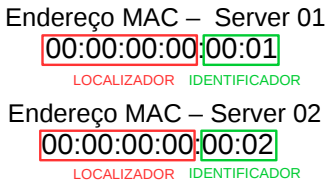


Figura 3. Endereçamento no hipercubo.

Para realizar a instalação de regras OpenFlow nas tabelas de encaminhamento dos nós utiliza-se a característica dos *bitmasks*, onde apenas parte dos endereços MAC é utilizado durante a busca por uma entrada na tabela que faça *matching* com um endereço de destino. Por exemplo, regras de encaminhamento para os endereços MAC 00:00:00:07:00:01 e 00:00:00:07:00:02, podem ser criadas com apenas 1 (uma) entrada, que será representado por 00:00:00:07:00:00/ff:ff:ff:ff:00:00. O funcionamento da máscara determina quais bits deverão ser utilizados durante o processo de verificação *matching*. Neste exemplo apenas os 32 primeiros bits da entrada na tabela deverão coincidir (*match*) com o endereço de destino para o qual se deseja encaminhar um pacote. Para o nó 000 da Figura 1(a) as entradas OpenFlow compactadas com REUNI e instaladas no plano de dados usando *bitmasks*, são apresentadas na Tab. 1.

Tabela 1. Exemplo das Regras com Bitmask no OF 1.3.

Seq.	Descrição da Regra	Ação	Interface
Entrada 1	prioridade=3, dl dst=00:00:00:04:00:00/ff:ff:ff:ff:00:00	output 4	100
Entrada 2	prioridade=2, dl dst=00:00:00:02:00:00/ff:ff:ff:ff:00:00	output 3	010
Entrada 3	prioridade=1, dl dst=00:00:00:01:00:00/ff:ff:ff:ff:00:00	output 2	001

4. Resultados e Discussões

Nesta seção o algoritmo de compactação das tabelas de encaminhamento é avaliado e os resultados são apresentados e discutidos. Para melhor organização esta seção foi subdividida em 4 (quatro) subseções, descritas a seguir.

A Seção 4.1 descreve as características do ambiente de experimentação e os procedimentos metodológicos de avaliação. Em seguida, na Seção 4.2, o algoritmo é avaliado em função da quantidade de entradas na tabela de encaminhamento de um nó em condições normais de funcionamento. Nesta etapa, surgiu então um questionamento relevante ao mecanismo proposto: dada a característica do algoritmo proposto de agregar vários destinos em um mesmo vizinho, será que a compactação das tabelas acarretaria em uma distribuição irregular da quantidade de fluxos que passam por cada um dos enlaces do hipercubo em uma comunicação de todos para todos?

Portanto, na Seção 4.3 avaliou-se a distribuição da quantidade de fluxos por enlace em um hipercubo completo com encaminhamento baseado em XOR. A avaliação foi efetuada levando-se em conta diferentes tamanhos de hipercubo a fim de se fazer um comparativo do comportamento da distribuição dos fluxos sem compactação (com encaminhamento XOR) e com compactação de tabelas (com encaminhamento OpenFlow). Por último, na Seção 4.4 foi efetuada uma avaliação da vazão em um hipercubo com 8 nós sem compactação de tabelas e com compactação de tabelas com o objetivo de avaliar o efeito da compactação da tabela e da redistribuição dos fluxos entre os enlaces no comportamento da vazão média dos fluxos no hipercubo.

4.1. Ambiente de Experimentação

Todo o protótipo foi emulado no Mininet, versão 2.2.1, tendo como máquina de encaminhamento o Open vSwitch [Pfaff et al. 2015] versão 2.3 e OpenFlow 1.3 [McKeown et al. 2008] como protocolo de comunicação entre os planos de dados e de controle. Para instalação das regras de encaminhamento no plano de dados utilizou-se o controlador Ryu, versão 3.23. Como ambiente de experimentação utilizou-se um *host* físico com sistema operacional Linux Ubuntu 14.04 64 bits, processador Core i5 3.2GHz, 6GB RAM. O algoritmo de compactação foi implementando utilizando-se o Python 2.7.

Para avaliar a vazão usou-se a ferramenta Iperf[2] versão 2.05. A configuração topológica ilustrada na Figura 1(a) representa um hipercubo de grau 3 (com 8 nós), usado como referência na maior parte dos experimentos.

4.2. Compactação das Tabelas de Encaminhamento

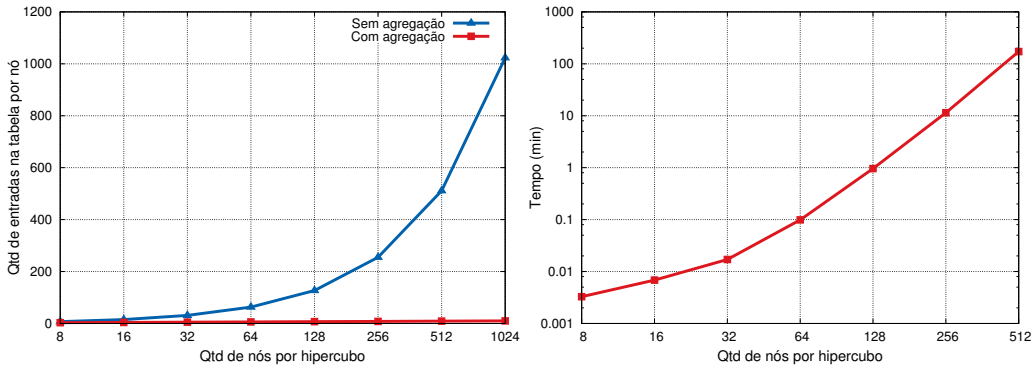
Neste experimento a quantidade de entradas nas tabelas de encaminhamento dos nós do hipercubo foi avaliada. O objetivo deste experimento é avaliar o impacto do algoritmo de compactação no tamanho das tabelas em cada nó. Desta forma avaliou-se a quantidade de entradas sem compactação e com compactação, usando o algoritmo proposto, em hipercubos completos de grau $\{3, 4, 5, 6, 7, 8, 9$ e $10\}$.

A Figura 4(a) apresenta os resultados obtidos neste experimento. Observe que à medida em que aumenta-se o tamanho do hipercubo há um incremento exponencial na quantidade de entradas na tabela de encaminhamento dos nós. Esse fato se deve à característica topológica, onde a quantidade de entradas ($QtdEntradas$) na tabela de um

[2]https://iperf.fr/

nó é igual a 2^n-1, onde n representa o grau do hipercubo. Ou seja, para um hipercubo de grau 10 tem-se: $QtdEntradas = 2^{10}-1 = 1024 - 1 = 1023$ entradas em cada nó da topologia. Não se considera, nesta avaliação, a possibilidade de existirem Máquinas Virtuais (VMs) ligadas aos nós do hipercubo, o que acarretaria mais entradas nas tabelas de encaminhamento dos nós sem compactação. Em contraste, a quantidade de entradas nas tabelas de encaminhamento permanece inalterada mesmo com a adição de VMs aos nós do hipercubo devido ao uso dos *bitmasks* do protocolo OpenFlow.

Se houver a utilização de VMs nos nós do hipercubo, no cenário sem compactação o número médio de entradas na tabela de encaminhamento em um nó será dado por $QtdEntradas = (2^{10}-1) \cdot m$, onde m corresponde ao número médio de VMs existentes em cada nó.



(a) (b)

Figura 4. Análise em função da quantidade de nós: (a) Número de entradas na tabela de encaminhamento de cada nó e (b) Tempo computacional para compactação.

Ainda na Figura 4(a) observa-se que com a compactação da tabela o incremento na quantidade de entradas é linear, onde tem-se: $QtdEntradas = n$. Em comparação ao exemplo anterior, para o hipercubo de grau 10 haverão apenas 10 entradas na tabela de encaminhamento, o que representa uma redução superior a 99% no número de entradas. A Figura 4(b) traz uma relação inicial do tempo necessário para compactação de todos os nós do hipercubo usando o REUNI. Observa-se que a complexidade da compactação é exponencial, mas é importante ressaltar que a ação de compactação pode ser feita *offline*, não acarretando prejuízo de desempenho para o *Data Center*.

4.3. Distribuição dos Fluxos entre os Enlaces

Conforme apresentado na Seção 3, a compactação da tabela de encaminhamento de um nó prioriza a agregação das entradas utilizando-se do conjunto completo dos k menores caminhos para cada destino. Contudo, dada a alta taxa de compactação do algoritmo proposto, especulou-se que tal resultado poderia impactar de forma negativa na distribuição dos fluxos nos enlaces do hipercubo. Por consequência, decidiu-se avaliar o comportamento da distribuição dos fluxos nos enlaces dos hipercubos de grau $\{3, 4, 5, 6, 7$ e $8\}$ durante a comunicação entre todos os pares origem e destino do hipercubo. Esta avaliação e a próxima (Seção 4.4) não foram realizadas para os graus $\{9$ e $10\}$ devido a limitações no hardware utilizado, o que estava ocasionando um alto tempo computacional para conclusão dos testes.

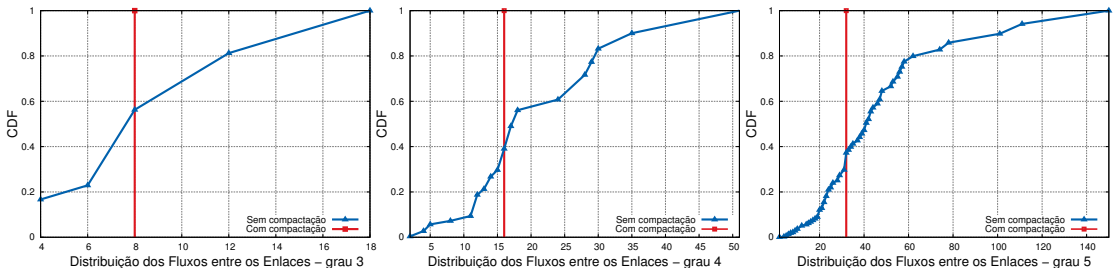
Os resultados são apresentados da seguinte maneira: é sabido que na topologia hipercubo todos os nós têm a mesma quantidade de enlaces incidentes, que é o grau do hipercubo (n). Dessa forma, a equação $QtdEnlaces = 2^{n-1} \cdot n$ define a quantidade total de enlaces em um hipercubo de grau n considerando-se todos os pares origem e destino. Além disso, para realizar a comunicação entre todos os pares origem e destino de um hipercubo de grau n produz-se um total de fluxos pode ser calculado pela Equação 1

$$QtdFluxos = \sum_{d=1}^{n} \frac{n!}{(n-d)d!} \cdot d \tag{1}$$

Conhecendo-se o total de fluxos, a distribuição ideal da quantidade de fluxos por enlace seria igual à média, portanto igual a $\frac{QtdFluxos}{QtdEnlaces} = 2^n$, que corresponde ao número de nós na rede. Desta forma, a distribuição ideal de fluxos para um hipercubo de grau 3 é de 8 fluxos por enlace, para um hipercubo de grau 4 é de 16 fluxos por enlace, e assim por diante.

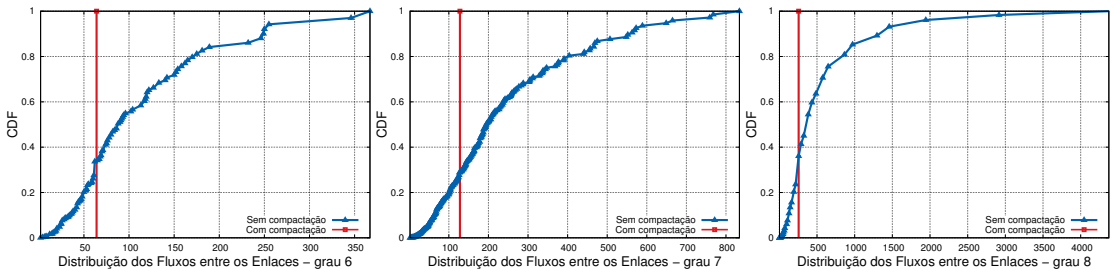
As Figuras 5 e 6 mostram a distribuição acumulada (CDF, ou *Cumulative Distribution Function*) da quantidade de fluxos por enlace sem compactação (usando encaminhamento XOR) e com compactação (segundo o algoritmo proposto), para hipercubos de grau $\{3, 4 \text{ e } 5\}$ e $\{6, 7 \text{ e } 8\}$, respectivamente. Como exemplo, na Figura 5(c) é possível observar um desbalanceamento na distribuição dos fluxos no cenário sem compactação (usando XOR), pois em mais de 60% dos enlaces existem mais fluxos do que o ideal. Entretanto, quando a compactação das tabelas de encaminhamento é utilizada, tem-se uma distribuição uniforme dos fluxos, ou seja, em 100% dos enlaces passam 32 fluxos na comunicação de todos para todos os nós do hipercubo.

Esse desbalanceamento natural dos fluxos usando XOR se repete em todos os casos avaliados. Em outro exemplo, no hipercubo de grau 4 apresentado na Figura 5(b), cerca de 61% dos enlaces passam mais de 16 fluxos no cenário sem compactação, ou seja, mais da metade dos enlaces teriam mais fluxos que o ideal. Esse resultado significa que em um total de 12 enlaces existem entre 17 e 51 fluxos. No total, 312 fluxos da comunicação de todos para todos seriam prejudicados por essa distribuição irregular sem a agregação das entradas. Contudo, com a compactação, novamente a distribuição é uniforme com 100% dos enlaces.



(a) (b) (c)

Figura 5. Análise da distribuição dos fluxos entre os enlaces no hipercubo: (a) com grau 3, (b) com grau 4 e (c) com grau 5.



(a) (b) (c)

Figura 6. Análise da distribuição dos fluxos entre os enlaces no hipercubo: (a) com grau 6, (b) com grau 7 e (c) com grau 8.

Na Figura 6(a) (grau 6), em cerca de 67%, (66 enlaces) trafegam mais fluxos do que a média em uma comunicação de todos para todos. Isso representa um total de 8152 fluxos disputando os 66 enlaces, com distribuição variando entre 65 e 367 fluxos por enlace. Contudo, após a compactação, além de reduzir a quantidade de entradas nas tabelas, observa-se uma distribuição equilibrada dos fluxos entre os enlaces do hipercubo, com exatamente 64 fluxos para cada um dos 192 enlaces. Esse comportamento repete-se nas Figuras 6(b) e 6(c).

Portanto, após avaliar o comportamento da distribuição dos fluxos, observou-se um comportamento completamente contrário ao esperado, ou seja, o algoritmo de compactação além de reduzir em escala logarítmica a quantidade de entradas nas tabelas de encaminhamento dos nós do hipercubo, também distribui os menores caminhos de maneira uniforme entre todos os enlaces da topologia. Esse resultado é devido ao fato de que o algoritmo considera todo o conjunto de menores caminhos entre todas as origens e destinos no hipercubo durante a compactação. Esse resultado leva a um balanceamento de carga natural entre os enlaces do hipercubo, facilitando dessa forma ações futuras de engenharia de tráfego em função das aplicações nos nós.

4.4. Comparativo da Vazão nos Enlaces com e sem Compactação das Tabelas

Neste experimento, o objetivo é avaliar o impacto da distribuição uniforme dos fluxos nos enlaces, ou seja, comprovar se essa uniformidade na distribuição dos fluxos beneficia a vazão em uma comunicação entre todos os pares origem e destino.

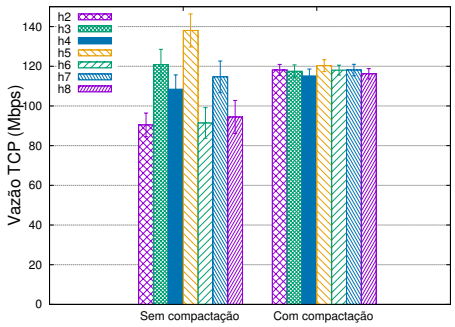


Figura 7. Comparativo da vazão (Mbps) com e sem compactação.

A Figura 7 apresenta a vazão de um *host* no nó $h1$ enviando simultaneamente tráfego para os outros 7 *hosts* em um hipercubo com a topologia da Figura 1(a) e enlaces de 1Gbps. Observa-se que após compactação das tabelas de encaminhamento de todos os

nós, a vazão também tende a ser uniforme. A pequena variabilidade apresentada é justificada devido ao mecanismo de controle de congestionamento do protocolo TCP. Neste experimento, todos os nós foram definidos como receptores de tráfego TCP gerado no Iperf e as medidas foram coletadas no cliente. Coletaram-se 30 amostras com duração de 10s cada amostra. A variabilidade foi obtida respeitando-se um intervalo de confiança de 95%. Esse comportamento se mantém para outros tamanhos de hipercubo, mas não foram apresentados aqui por limitações de espaço.

De acordo com os resultados obtidos nos experimentos, a distribuição uniforme dos fluxos aumenta a vazão média do hipercubo em cerca de 8%, além de apresentar uma menor variabilidade e, consequentemente, prover uma comunicação mais estável e com menor *jitter*.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou o REUNI, uma proposta de algoritmo que reduz em mais de 99% a quantidade de entradas nas tabelas de encaminhamento e uniformiza a distribuição dos fluxos, em uma comunicação de todos para todos, em 100% dos enlaces, ambos no plano de dados. Uma descrição detalhada do algoritmo foi apresentada e a proposta fora avaliada com hipercubos de grau $\{3, 4, 5, 6, 7, 8\}$, contribuindo para facilitar a adoção dessa topologia em *Data Centers* centrados nos servidores, dado que o crescimento exponencial das tabelas de encaminhamento era um problema importante nessa topologia. Foi implementado um protótipo usando o Mininet, controlador Ryu e OpenFlow 1.3 como prova de conceito.

O REUNI alcança uma taxa de compactação logarítmica das tabelas de encaminhamento dos nós, que passam a ter $log_2(2^n) = n$ entradas, independentemente da existência de virtualização de servidores no *Data Center*. Em todos os casos avaliados observou-se que a distribuição dos fluxos utilizando operações XOR apresenta um desbalanceamento na distribuição da carga de tráfego entre os enlaces. No pior caso, para o hipercubo de grau 7, cerca de 71% dos enlaces ficam sobrecarregados com mais de 128 fluxos, totalizando 40853 fluxos distribuídos em 159 enlaces. Os resultados de desempenho apresentados neste artigo, efetuados em um ambiente emulado, indicam que a distribuição uniforme dos fluxos entre os enlaces realizada pelo REUNI é capaz de melhorar a vazão média do hipercubo em cerca de 8%.

Os próximos passos no desenvolvimento deste trabalho serão: i) otimizar o processo de agregação das regras de entrada da tabela. Para isso, estamos trabalhando para reduzir e/ou concatenar etapas do algoritmo proposto, o que impactará na redução do tempo computacional; ii) integrar o REUNI com o FlexForward [Vencioneck et al. 2014], implementado no Open vSwitch, o que permitirá aumentar a vazão em cada nó do hipercubo e a mudança automática da forma de encaminhamento dos pacotes em caso de falhas; iii) avaliar a implementação no *Data Center* do NERDS (Núcleo de Estudos em Redes Definidas por Software, UFES), em ambiente real.

Durante os experimentos foi descoberto que o uso dos *bitmasks* no Open vSwitch força o uso do espaço do usuário (*user space*) no encaminhamento dos pacotes, o que limita a vazão de encaminhamento. Superar essa limitação do Open vSwitch (e não do protótipo) também é um possível trabalho futuro.

Referências

Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74.

Braun, W. and Menth, M. (2014). Wildcard compression of inter-domain routing tables for openflow-based software-defined networking. In *Software Defined Networks (EWSDN), 2014 Third European Workshop on*, pages 25–30.

Dally, W. (1990). Performance analysis of k-ary n-cube interconnection networks. *Computers, IEEE Transactions on*, 39(6):775–785.

de Lima, D. S. A., Guimarães, R. S., Vassoler, G. L., Liberato, A. B., Martinello, M., and Villaca, R. S. (2015). Avaliação do uso do openflow na recuperação de falhas em data centers centrados nos servidores. In *Anais do XIV Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, WPerformance '15, Recife/PE, Brasil. SBC.

Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., and Lu, S. (2009). BCube: A High Performance, Server-Centric Network Architecture for Modular Data Centers. *SIGCOMM Comput. Commun. Rev.*, 39(4):63–74.

Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., and Lu, S. (2008). Dcell: A scalable and fault-tolerant network structure for data centers. *SIGCOMM Comput. Commun. Rev.*, 38(4):75–86.

Luiz Fernando T. de Farias, Morganna C. Diniz, S. C. d. L. (2014). Uma abordagem para redução da tabela de encaminhamento sob a ótica da interface de saída dos pacotes. In *XIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance 2014)*, Porto Alegre/RS. SBC.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., and Casado, M. (2015). The design and implementation of open vswitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 117–130, Oakland, CA. USENIX Association.

Saad, Y. and Schultz, M. H. (1989). Data communication in hypercubes. *Journal of Parallel and Distributed Computing*, 6(1):115 – 135.

Vassoler, G. L. (2015). *TRIIIAD: Uma Arquitetura para Orquestração Automônica de Redes de Data Center Centrado em Servidor.* PhD thesis, Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo (PPGEE/UFES), Vitória/ES, Brasil.

Vencioneck, R., Vassoler, G., Martinello, M., Ribeiro, M., and Marcondes, C. (2014). Flexforward: Enabling an sdn manageable forwarding engine in open vswitch. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 296–299.

Trilha Principal do SBRC 2016
Sessão Técnica 14
Segurança em Redes

Um Sistema Acurado de Detecção de Ameaças em Tempo Real por Processamento de Fluxos*

Antonio Gonzalez Pastana Lobato,
Martin Andreoni Lopez,
Otto Carlos Muniz Bandeira Duarte

[1]Grupo de Teleinformática e Automação – GTA
Universidade Federal do Rio de Janeiro – UFRJ
Rio de Janeiro – RJ – Brasil

{antonio, martin, otto}@gta.ufrj.br

Abstract. *The late detection of security threats causes a significant increase of the risk of irreparable damages, disabling any defense attempt. Although very hard to analyze, attacks always leave traces that can be detected. This paper proposes a real time streaming threat detection system based on machine learning algorithms. The system architecture combines the advantages of real time stream processing with the batch processing over a historical database that does not require manual interaction of security specialists. The proposed system allows anomaly-based detection of known and zero-day attacks. The system was developed and evaluated from a data set constructed with the capture of legitimate and malicious network traffic. Results show that the proposed system presents an accurate threat detection with low processing time, permitting prompt defense strategies.*

Resumo. *A detecção tardia de ameaças provoca o aumento substancial dos riscos de danos irreparáveis e inviabiliza qualquer tentativa de defesa. Embora sejam difíceis de serem descobertos, os ataques deixam vestígios que podem ser detectados. Este artigo propõe um sistema de detecção de ameaças em tempo real por processamento de fluxos baseado em algoritmos de aprendizado de máquinas. A arquitetura do sistema combina as vantagens do processamento em tempo real de fluxos com as do processamento em lotes sobre uma base histórica e não requer a intervenção de especialistas de segurança para a análise e readaptações às ameaças. O sistema proposto permite tanto a detecção de ataques conhecidos quanto de novos ataques, ainda desconhecidos, através de métodos automatizados de classificação de ataques e de detecção de anomalias. O sistema foi desenvolvido e avaliado a partir de um conjunto de dados construído com a captura de tráfego de rede legítimo e malicioso. Os resultados mostram que o sistema apresenta uma acurada detecção de ameaças com baixo tempo de processamento, o que possibilita estratégias de defesa.*

1. Introdução

Os riscos e os ataques de segurança são atualmente muito altos e tendem a aumentar significativamente no futuro com a introdução da Internet das coisas (*Internet of Things*- IoT), uma vez que estima-se em mais de 80 bilhões os dispositivos interconectados até 2025 [Clay 2015]. Esse cenário exibe uma alta complexidade no gerenciamento e na proteção das redes de comunicação, gerando desafios na segurança e na privacidade dos dados. Os bilhões de dispositivos proveem grandes massas de dados (*Big Data*) [Costa et al. 2012], gerados em forma

*Este trabalho foi realizado com recursos da CNPq, CAPES, FAPERJ e FINEP.

de fluxos, que precisam ser gerenciados, processados, transferidos e armazenados de forma segura em tempo real [Porto e Ziviani 2014]. Esse cenário introduz novos desafios tecnológicos, uma vez que as tecnologias atuais não foram projetadas para essa demanda. Além disso, a virtualização, amplamente utilizada em sistemas de computação em nuvem, introduz novas ameaças. Assim, tanto o uso da tecnologia de virtualização quanto a velocidade, o volume e a variedade das grandes massas de dados são fatores que geram novas vulnerabilidades.

O tempo de detecção de uma ameaça é crucial para manter a segurança nos sistemas de comunicação [Wu et al. 2014]. A detecção de ameaças de forma eficiente exige o monitoramento, o processamento e o gerenciamento das grandes massas de dados, que permitem extrair informações da caracterização do tráfego. No entanto, detectar ameaças em grandes massas de dados requer o desenvolvimento de técnicas modernas de análise. Os atuais sistemas projetados para coletar dados de diferentes fontes e gerenciar em um ponto único as informações de segurança, como o *S*ecurity Information and Event Management (SIEM), não são eficazes, pois 85% das intrusões na rede são detectadas depois de semanas de sua ocorrência [Ponemon e IBM 2015]. Além disso, a reação às ameaças é muito lenta, em média 123 horas depois de ocorridas, e a detecção no vazamento de informações demora em média 206 dias [Clay 2015]. Portanto, o longo tempo de detecção dessas ameaças impossibilita qualquer tipo de defesa. Logo, as técnicas de analítica[1] para processamento de fluxo em tempo real permitem o imediato processamento e análise de diferentes tipos de dados e, consequentemente, a detecção de ameaças. Um outro problema conhecido é a enorme quantidade de alarmes gerados pelos sistemas atuais de monitoramento. Esses alertas levam a falsos positivos, que são, na maior parte, ignorados por analistas de segurança, junto com eventuais reais positivos que representam graves ameaças, por não conseguirem lidar com a quantidade de dados que chegam.

Este artigo propõe e desenvolve um sistema para a detecção de ameaças em tempo real, utilizando diversas plataformas de código aberto. A integração das diversas plataformas permite a análise de grandes volumes de dados pelo processamento por fluxo (*stream processing*). O sistema proposto utiliza técnicas de aprendizado de máquina para a classificação de ataques e a detecção de anomalias. Além disso, para a avaliação do sistema, foi criado um conjunto de dados com as classes marcadas, contendo o uso normal e ataques, a partir da captura de tráfego de rede, abstraídos em fluxos. A arquitetura do sistema utiliza o conceito da arquitetura lambda [Marz e Warren 2013], na qual é combinado o processamento tradicional em lotes (*batch*) sobre uma base histórica, com o processamento em tempo real por análise de fluxos. Assim, o sistema proposto é capaz de detectar tanto ataques conhecidos quanto os novos ataques, ainda desconhecidos, através de métodos automatizados de classificação de ataques e de detecção de anomalias. Um protótipo do sistema foi desenvolvido e avaliado. Os resultados mostram uma alta acurácia do sistema desenvolvido para detectar ameaças. Além do sistema proposto detectar rapidamente ameaças de segurança, o processamento de fluxos é associado ao processamento em tempo diferenciado de dados armazenados historicamente para adaptar os algoritmos de detecção em tempo real. Isto resulta em um sistema com um valor baixo de falsos positivos, possibilitando a implementação de estratégias de defesa.

O restante do artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. O sistema proposto é apresentado na Seção 3. O conjunto de dados para a avaliação do sistema é apresentado na Seção 4. Na Seção 5 são discutidos os métodos para a detecção de ameaças e os resultados obtidos. Por fim, a Seção 6 conclui o artigo.

[1]Analítica é a ciência da análise lógica, isto é, a análise de enormes conjuntos de dados, usando matemática, estatística e software de computação, para achar padrões de dados que proveem conhecimento, informação útil ou visões de dados em bruto.

2. Trabalhos Relacionados

No aprendizado de máquina existem diversas técnicas ou algoritmos que podem ser classificadas de acordo com o procedimento aplicado em supervisionadas ou não-supervisionadas. Na análise supervisionada o conjunto de dados é totalmente etiquetado. Este tipo de análise é usado para a classificação de ataques. Entre os exemplos mais conhecidos de técnicas usadas na classificação estão as redes neurais, as árvores de decisão e as máquinas de vetores de suporte (*Support Vector Machine* - SVM) [Buczak e Guven 2015, Stroeh et al. 2013]. Esses artigos utilizam o conjunto de dados KDD 99, um dos poucos conjuntos de dados que existem na área de pesquisa relacionada à detecção de Intrusão. Na análise não-supervisionada, o conjunto de dados é utilizado sem informação dos tipos de classes às quais eles pertencem. Este tipo de análise é aplicado na detecção de padrões. No entanto, nenhuma destas técnicas foi ainda aplicada em tempo real.

O Snort e o Bro são as ferramentas de *Software* Livre mais populares que realizam a Detecção de Intrusão em tempo real [Rai e Devi 2013]. O Snort utiliza apenas o método de detecção por assinatura, não detectando pequenas variações dos ataques e requer atualizações constantes na base de dados das assinaturas. O Bro, por sua vez, apresenta uma linguagem para a detecção por anomalias, mas as técnicas para a detecção devem ser criadas pelo próprio usuário. Estas ferramentas podem ser usadas para melhorar a prevenção de ataques, em conjunto com outros mecanismos, como as redes definidas por *software* (*Software Defined Networks*-SDN), as quais são usadas para realizar a detecção e prevenção de intrusão em redes virtuais [Andreoni Lopez et al. 2014, Lobato et al. 2014].

Holtz *et al.* propõem um sistema de detecção de intrusão para grandes massas de dados utilizando técnicas de aprendizado de máquinas *Map-Reduce* [Holtz et al. 2011]. Embora a técnica do *Map-Reduce* seja factível para analisar grandes massas de dados, ela não possui bom desempenho para detecção de ameaças em tempo real.

Outro aspecto importante abordado por diversos pesquisadores é a tarefa de criação de conjuntos de dados para analisar, avaliar e comparar propostas de sistemas de detecção de ameaças. WebClass [Ringberg et al. 2008] é uma ferramenta *web* que permite a contribuição manual dos usuários na etiquetagem do tráfego de redes. No entanto, esta proposta não chamou a atenção dos pesquisadores e o projeto foi abandonado. Sperotto *et al.* [Sperotto et al. 2009] criaram um conjunto de dados com mais de 14.2 milhões de fluxos etiquetados. Algumas pesquisas criaram seus conjuntos de dados para avaliar as suas propostas [Amini et al. 2006, Sangkatsanee et al. 2011]. Sangkatsanee *et al.* utilizam diferentes algoritmos para realizar a classificação de ataques, enquanto Morteza Amini *et al.* detectam anomalias em tempo real utilizando redes neurais não-supervisionadas. Estes artigos não avaliam suas propostas em cenários que apresentem uma topologia com um grande número de máquinas com seu tráfego sendo analisado.

O SAND [Liu et al. 2014] é uma ferramenta para o monitoramento de redes através do processamento por fluxos, no entanto esta proposta só caracteriza o tráfego de redes sem a detecção de ataques. A detecção de anomalias no desempenho de máquinas virtuais usando técnicas de inteligência de máquinas é proposta em [Solaimani et al. 2014]. O trabalho [Du et al. 2014] não apresenta possibilidade de análises de dados provenientes de diferentes fontes, e no trabalho de Zhao *et al.* [Zhao et al. 2015] os resultados ainda são preliminares para obtenção de maiores conclusões.

Diferentemente dos artigos citados anteriormente, este artigo propõe o uso de plataformas de software aberto em uma arquitetura específica que permite o processamento e a

análise de fluxos de dados em tempo real ao mesmo tempo que se apoia em uma base de dados histórica.

3. O Sistema Proposto

A análise de grandes conjuntos de dados estáticos é comumente realizada por processamento em lotes (*batch*). No entanto, esta técnica apresenta grandes latências, com respostas superiores a 30 segundos, enquanto algumas aplicações requerem processamento em tempo real com respostas da ordem de sub-segundo [Rychly et al. 2014]. Por sua vez, a técnica de processamento de fluxo de dados (*streaming processing*) analisa uma sequência massiva de dados ilimitados que são continuamente gerados. Estes paradigmas são combinados na arquitetura lambda para obter analíticas de grandes massas de dados em tempo real.

O sistema proposto é baseado na arquitetura lambda, mostrada na Figura 1, que permite a manipulação e a análise em tempo real de quantidades massivas de informação. A arquitetura lambda [Marz e Warren 2013] é composta por três camadas: a camada de processamento de fluxo de dados, a camada de processamento em lotes e a camada de serviço. A camada de processamento de fluxo de dados trata o fluxo de dados em tempo real. A camada de processamento em lotes analisa grande quantidade de dados armazenados através de técnicas de computação distribuída como *map-reduce*. Finalmente, a camada de serviço agrega as informações obtidas das duas camadas anteriores para criar as saídas dos dados analisados. Assim, o objetivo da arquitetura lambda é analisar, em tempo real e de forma acurada, os fluxos de dados e *logs* de diferentes fontes nas velocidades que são gerados.

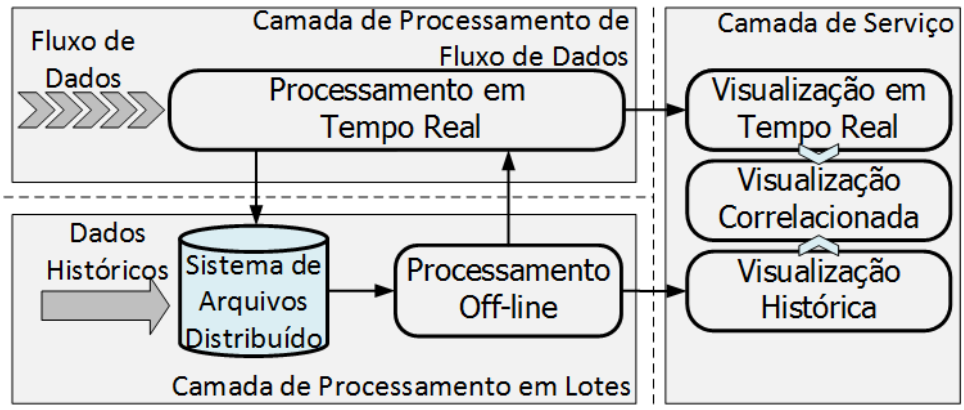


Figura 1: As três camadas da arquitetura *lambda* de processamento simultâneo em *tempo real* e em tempo diferenciado: processamento de fluxo de dados, processamento em lotes (batch) e serviço.

A análise dos dados é dividida em três etapas: captura, normalização e processamento. Primeiro, os dados são capturadas no local de coleta. Em seguida, todas essas informações adquiridas são enviadas para o processo de normalização, onde os dados são formatados e enriquecidos a altíssimas taxas de processamento com informações externas, tais como informações geográficas, correlações, etc. Assim, cria-se uma qualidade mais rica de informação sobre um ataque. Com a correlação e o enriquecimento de dados de distintas fontes, ao se detectar uma ameaça, é possível responder a perguntas, tais como: quem foi o atacante, a sua localização geográfica, os locais dos quais as informações vazaram, o destino para o qual as informações foram enviadas, entre outras. Finalmente, os dados normalizados são processados em tempo real, adquirindo os padrões necessários para a segurança. Neste processo, uma vez obtidos os resultados, eles são visualizados e armazenados.

O protótipo de detecção de ameaças do sistema proposto é composto de ferramentas de *software* de código aberto. A configuração do sistema consiste na integração das seguintes ferramentas de código aberto:

Bro[2]**:** é uma ferramenta de monitoramento e análise em tempo real do tráfego de rede. O `Bro` possui uma linguagem de programação orientada a redes, o que facilita a abstração dos fluxos;

Flume[3]**:** é um serviço distribuído para a coleta, agregação e movimentação de grandes quantidades de dados em formato de *logs*;

Kafka[4]**:** é um mediador de mensagens (*Broker*) que funciona como um serviço de Produtor/Consumidor. O `Kafka` abstrai o fluxo de mensagens em tópicos. Os Produtores gravam os dados em tópicos e os consumidores leem os dados a partir desses tópicos;

Storm[5]**:** é um processador de eventos em tempo real , também conhecido como *Data Streaming Processor* (DSP). O `Storm` é a ferramenta central da proposta. Este processador é composto por topologias que formam Grafos Acíclicos Dirigidos (GAD) compostos por elementos de entradas, os *spouts*, e elementos de processamento, os *bolts*. É possível usar o paralelismo dos *spouts* e dos *bolts*, fazendo com que as diferentes amostras dos fluxos possam ser processadas em tempo real e de forma paralela. Uma topologia funciona como um canal de dados, que são processados em forma de fluxo à medida que os dados avançam;

HBase[6]**:** é uma base de dados distribuída que armazena dados não-relacionados. O `HBase` não suporta pedidos *query* do tipo SQL, como ocorre nas bases de dados estruturadas. No entanto, ela permite o armazenamento de grandes massas de dados dispersos e o acesso aos dados em tempo real.

A Figura 2 mostra como os programas de código aberto são interligados para compor o sistema proposto. A coleta de dados é obtida de diferentes fontes. Essas fontes são *logs* do sistema operacional e também de aplicações, além dos dados de fluxo de redes que são coletados do `Bro` em diferentes pontos da rede. Logo, no sistema proposto, o `Bro` atua como um caracterizador de fluxos, sintetizando as informações dos pacotes que entram pelas interfaces de rede do sistema e agrupando as informações em janelas de tempo.

O sistema visa garantir a segurança de diversos componentes da rede. Para isso, as coletas de dados ocorrem em locais distintos da rede e são agrupadas para serem processadas. Tanto as informações de fluxo geradas pela ferramenta `Bro`, como os *logs* das aplicações, são transportados até o local de análise pela ferramenta Apache `Flume` que atua como um túnel ligando as fontes de dados à parte de processamento do sistema. Como a geração de dados de segurança do sistema possui uma taxa variável em relação ao uso, pode haver momentos em que a plataforma de processamento não consiga processar todos os dados imediatamente. Por isso, os dados de segurança são recebidos pela ferramenta Apache `Kafka` no local de análise. Basicamente, esta ferramenta serve como um *buffer* para a ferramenta de processamento, adequando taxas distintas de geração e processamento. O Apache `Storm` é o núcleo de processamento em tempo real da plataforma, para detecção de ameaças. O `Storm` oferece um método de computação processamento de fluxos distribuído e tolerante falhas. Assim, o `Storm` realiza o processamento em memória, garantindo baixa latência em tempo real.

[2]Bro IDS: https://www.bro.org/
[3]Apache Flume: https://flume.apache.org/
[4]Apache Kafka: http://kafka.apache.org/
[5]Apache Storm: http://storm.apache.org/
[6]Apache HBase: https://hbase.apache.org/

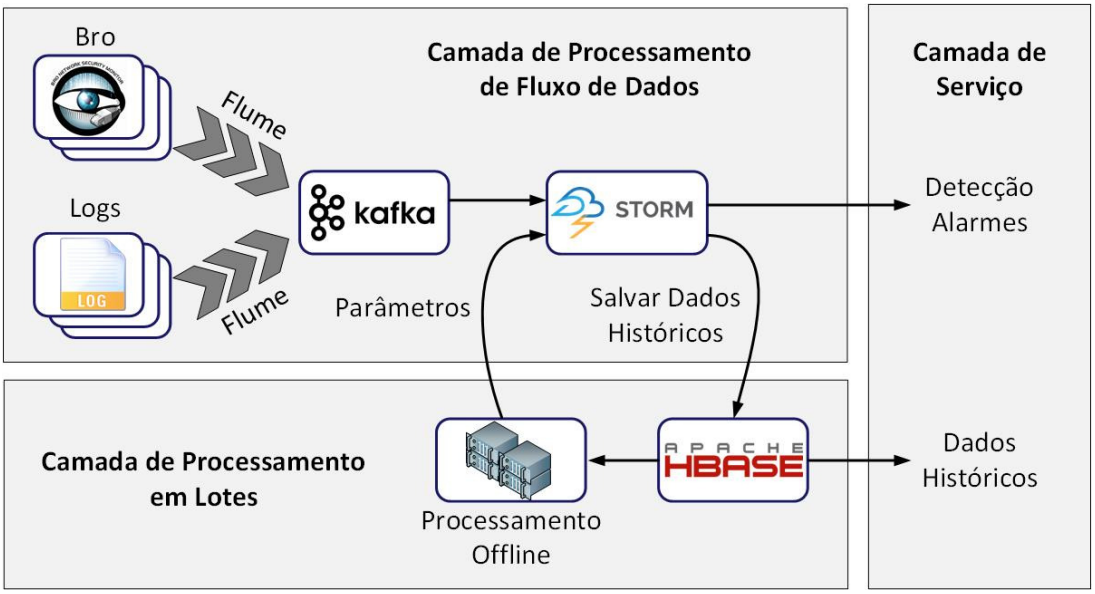


Figura 2: Exemplo de configuração do sistema proposto para análise de dados por processamento de fluxos e processamento de lotes com uso de ferramentas de *Software* livre.

Uma vez que as analíticas são extraídas dos dados por fluxos, os resultados são armazenados para uma análise posterior em uma base de dados dinâmica. A base de dados utilizada na proposta é a Apache `HBase`. Os dados armazenados contêm as informações obtidas da detecção e podem ser processados de maneira *off-line* a calcular parâmetros que são utilizados no modelo em tempo real. Neste ponto existe uma realimentação do sistema, já que os parâmetros calculados no processamento *off-line* com os dados históricos servem para ajustar o modelo de processamento em tempo real. Dessa forma, o sistema possui uma característica adaptativa, pois os parâmetros podem ser atualizados, se ajustando a novos padrões de uso.

4. Criação do Conjunto de Dados

Poucos conjuntos de dados (*datasets*) de segurança de redes são públicos para a avaliação de mecanismos de detecção de ataques. O principal motivo para a não liberação de dados de segurança é a privacidade, pois o tráfego de rede pode conter informações confidenciais. Os dois principais conjuntos de dados disponíveis são o DARPA [Lippmann et al. 2000] e o KDD 99 [Lee et al. 1999]. O DARPA foi criado com dados de tráfego (TCP/IP) e informações de sistema operacional obtidos através de uma rede de computadores simulada. Durante a coleta dos dados, também foram simulados ataques e marcados no conjunto de dados. Já o KDD 99 foi feito a partir da seleção e agrupamento de características do DARPA. Contudo, esses conjuntos de dados apresentam várias limitações [Tavallaee et al. 2009]. Uma delas é que o tráfego não corresponde ao cenário de uma rede real, por ser simulado. Além disso, há dados redundantes, o que influencia nos resultados dos algoritmos. Outro problema é que esses conjuntos de dados possuem mais de 15 anos e não representam ataques atuais [Sommer e Paxson 2010].

Uma contribuição deste trabalho é a criação de um conjunto de dados com tráfego real de rede para a avaliação da proposta[7]. Seguindo o mesmo tipo de procedimento usado na elaboração do conjunto de dados DARPA, um conjunto foi elaborado a partir da captura de pacotes em máquinas do laboratório do GTA/UFRJ, contendo tanto tráfego normal das máquinas, quanto ataques reais de segurança. Após a captura dos pacotes, as informações de fluxo foram

[7]O conjunto de dados será disponibilizado contactando os autores por *e-mail*.

obtidas por meio da extração de dados dos cabeçalhos dos pacotes, agrupadas em janelas de 2 segundos, pois dessa forma agrega-se informação suficiente para a composição dos fluxos. Os fluxos são definidos pela sequência de pacotes de um mesmo IP de origem para um mesmo IP de destino. Para cada um dos fluxos, foram obtidas 24 características a partir dos cabeçalhos TCP/IP dos pacotes de cada fluxo, como por exemplo: taxa de pacotes TCP, UDP e ICMP; quantidade de portas de destino e origem; quantidade de cada um dos *flags* TCP; entre outras. Existem duas classes de ameaças que podem ser detectadas pela análise das informações dos cabeçalhos, os ataques de negação de serviço (*Denial of Service* - DoS) e varredura de portas. Logo, diversos tipos de ataques dessas duas classes foram feitos para formar o conjunto de dados. Ao todo, o conjunto de dados contém 7 tipos de DoS e 9 tipos de varredura de portas distintos. Os ataques de DoS feitos são: inundação ICMP, *land*, *nestea*, *smurf*, inundação SYN, *teardrop* e inundação UDP. Já os tipos de varredura de portas no conjunto de dados são: *TCP SYN scan*, *TCP connect scan*, *SCTP INIT scan*, *Null scan*, *FIN scan*, *Xmas scan*, *TCP ACK scan*, *TCP Window scan* e *TCP Maimon scan*. Todas as ameaças foram realizadas com ferramentas da distribuição *Kali Linux*[8], que é voltada para a segurança de computadores. Esses ataques foram marcados no conjunto de dados através de filtros de origem e destino que permitem separar o tráfego das máquinas atacantes das demais.

Ao todo, cerca de 95 GB de dados de captura de pacotes foram coletados, resultando em 154.187 fluxos entre ataques e tráfego normal. Para a classificação, o conjunto de dados foi dividido em 70% para treino e 30% para avaliação do desempenho dos algoritmos implementados, o que faz com que as estatísticas sejam obtidas em dados novos, não analisados na parte de treino, como é comumente usado nos trabalhos científicos da área. Já para a detecção de anomalia, o treino é feito com 70% dos dados de tráfego legítimo para a determinação do comportamento normal. Os outros 30% são usados para calcular a taxa de falsos positivos e os dados de ataque são usados para calcular a taxa de detecção.

4.1. Seleção de Características e Análise de Componentes Principais

Para aumentar a eficiência do sistema proposto na detecção de ameaças em tempo real, optou-se pela redução da dimensionalidade. Busca-se a eliminação de algumas características irrelevantes para a classificação de ataques, tais como as características que não variam de valor ou com uma distribuição uniforme, que pouco contribuem sobre o pertencimento de uma amostra a uma classe específica. Outro aspecto que deve ser considerado é uma possível correlação entre duas ou mais características, que poderiam ser combinadas em apenas uma característica, diminuindo o tempo de processamento.

A redução de dimensionalidade é obtida pelo algoritmo de Análise de Componentes Principais (*Principal Component Analysis* - PCA), que transforma um grupo de variáveis possivelmente correlacionadas em um grupo de variáveis linearmente descorrelacionadas, que estão em planos ortogonais. Essa transformação é feita levando em conta a decomposição em autovalores, de maneira que o componente associado ao maior autovalor represente a maior variância dos dados. Os demais componentes da matriz resultante também são ordenados de acordo com a variância que eles representam. Assim, as componentes associadas aos autovalores maiores possuem mais informação relevante, fazendo com que os associados aos mais baixos possam ser retirados, reduzindo a dimensionalidade dos dados e diminuindo o tempo necessário para os algoritmos. Deve ser ressaltado que o algoritmo de análise de componentes principais não leva em conta as marcações de classe para realizar a transformação dos dados e, portanto, foi usado tanto nos algoritmos supervisionados, quanto nos algoritmos não supervisionados de aprendi-

[8]Linux Kali: https://www.kali.org/

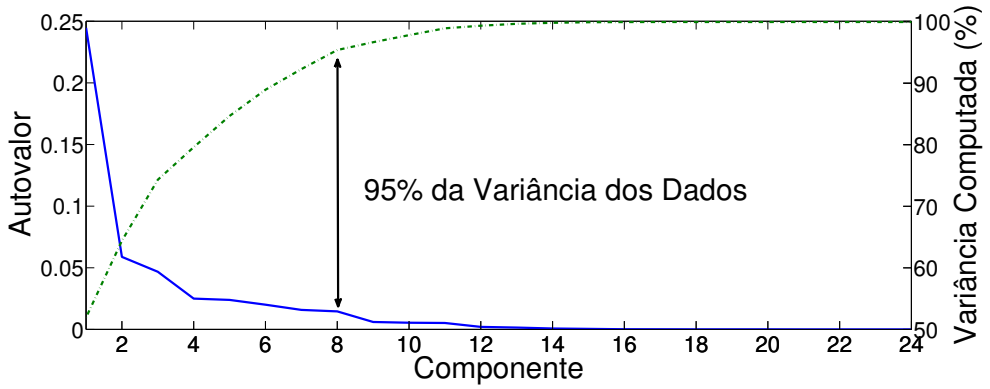


Figura 3: Autovalor para cada uma das 24 características dos fluxos do conjunto de dados. O autovalor associado à cada característica está diretamente ligado à variância do conjunto de dados. Os oito maiores componentes principais representam 95% da variância do conjunto completo com as 24 características.

zado de máquinas.

A Figura 3 mostra os autovalores associados ao conjunto de dados elaborado para avaliar o sistema proposto. A soma dos 8 maiores autovalores representa 95% da soma total, o que equivale a dizer que as 8 primeiras características dos dados resultantes da transformação linear calculada pela PCA representam 95% da variância total. Logo, essas 8 componentes são selecionadas e as outras componentes, que representam apenas 5% da variância, são descartadas, diminuindo assim o tempo de processamento, que é crítico em aplicações de tempo real.

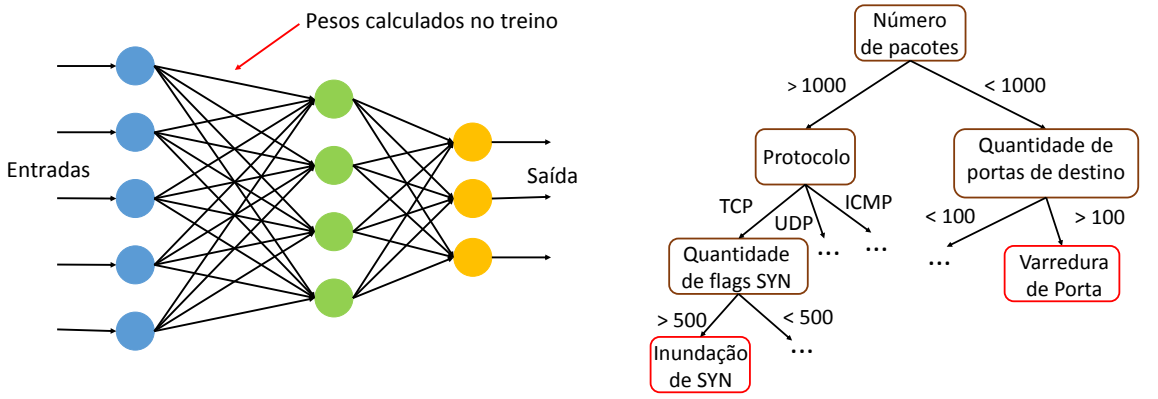
5. A Detecção Automática de Ameaças

Uma importante característica do sistema proposto para a obtenção da detecção em tempo real, é a eliminação completa da intervenção de especialistas de segurança para classificar e para configurar. O sistema proposto se baseia em algoritmos supervisionados de aprendizado de máquinas, como por exemplo os apresentados na Figura 4, para realizar a classificação automatizada de ameaças. Desta forma, fica sob responsabilidade dos algoritmos descobrir as características de cada classe de ataque, ao invés de ter a configuração manual como nos sistemas de segurança atuais. Além disso, a realimentação dos dados de fluxo para o processamento *off-line*, apresentada na Figura 2, permite que o sistema apresente atualizações dinâmicas, possuindo um comportamento adaptativo.

Nas Seções 5.1, 5.2 e 5.3 são apresentados os algoritmos de classificação implementados para a avaliação do sistema. Em todos os métodos são usados 70% do conjunto de dados para treino e 30% para teste e, durante a fase de treino, também é feita a validação cruzada para evitar a especialização. Na validação cruzada, partes dos dados de treino são separadas e não são usadas no cálculo dos parâmetros do modelo. Elas são usadas posteriormente para verificar se o modelo está geral o suficiente para se adaptar a novos dados, e não especializado somente para os dados de treino.

5.1. O Algoritmo de Árvore de Decisão

Nas árvores de decisão, as folhas representam a classe final e os ramos representam condições baseadas no valor de uma das características de entrada. Durante a fase de treino foi determinada a estrutura da árvore de classificação, usando o algoritmo de aprendizado C4.5. Na implementação do algoritmo de árvore de decisão para uso em tempo real, são usados os



(a) Esquema de algoritmo de rede neural. **(b) Esquema de algoritmo de árvore de decisão.**

Figura 4: Algoritmos de classificação: a) na rede neural, cada neurônio executa uma parte do processamento b) na árvore de decisão, as características são analisadas e no fim é obtida uma classificação.

parâmetros calculados durante a fase de treino, na qual são determinadas regras que geram um grafo semelhante ao da Figura 4b, com a diferença que as características avaliadas são as obtidas após a seleção de características com a PCA. Portanto, para a implementação no sistema em tempo real, essas regras são escritas no formato de expressões condicionais (*if-then-else*) que resultam no grafo determinado na fase de treino. Os resultados são apresentados na Seção 5.4 junto com os demais algoritmos de classificação.

5.2. O Algoritmo de Rede Neural

A implementação do algoritmo de rede neural segue o funcionamento do cérebro humano, em que cada neurônio é responsável por uma parte do processamento, passando o resultado para o neurônio seguinte. Na saída, é obtido um grau de pertinência a cada classe, sendo que a classe escolhida é a de maior grau. Os vetores de peso Θ são calculados durante o treino. Na Figura 4a esses pesos são representados pelas ligações entre os neurônios. Primeiro obtém-se um espaço amostral de entradas e saídas desejadas para a rede neural e posteriormente minimiza-se a parcela do erro da estimativa de cada parâmetro através do algoritmo de *Back Propagation*.

O cálculo realizado por cada camada para rede neural para determinar a classe que a amostra pertence é realizado através das seguintes equações:

$$z_{(i+1)} = \Theta_{(i)} a_{(i)} \qquad (1) \qquad a_{(i+1)} = g(z_{(i+1)}) \qquad (2) \qquad g(z) = \frac{1}{1+e^{-z}} \qquad (3)$$

onde $a_{(i)}$ é o vetor que determina a saída correspondente à camada i, $\Theta_{(i)}$ é o vetor de pesos que leva da camada i até a camada $i+1$, e $a_{(i+1)}$ representa a saída da camada seguinte, a camada $i+1$. A função $g(z)$ é a função *Sigmoid* e ela desempenha um papel muito importante na classificação. Para valores elevados de z, ela vale 1 e para valores baixos, ela vale 0. Portanto, na última camada é obtida um grau de pertencimento de cada classe, entre 0 e 1, sendo a classe escolhida a de maior grau de pertencimento.

5.3. O Algoritmo de Máquina de Vetores de Suporte

A Máquina de Vetores de Suporte (*Support Vector Machine* - SVM) é um classificador binário baseado no conceito de planos de decisão que definem limiares de decisões. Basica-

mente, o SVM classifica através da construção de hiperplanos em um espaço multidimensional que separa casos de diferentes rótulos de classe. Um algoritmo de treinamento iterativo é usado para minimizar uma função de erro, construindo um hiperplano ótimo. O algoritmo de SVM separa os dados de treinamento em espaço de características por um hiperplano definido pelo tipo de função de *kernel* utilizado. Assim, SVM encontra o hiperplano de margem máxima, definida como a soma das distâncias do hiperplano desde o ponto mais próximo das duas classes.

A detecção em tempo real é realizada pela classificação para cada uma das classes, sendo normal e não normal; *DoS* e não *DoS*; e varredura e não varredura. Uma vez obtida a saída, é escolhida a classe que obtém a maior pontuação. A pontuação para classificar uma observação x, é a distância desde x aos limiares de decisão variando de $-\infty$ para $+\infty$. A pontuação é computada como a resposta a função:

$$f(x) = \sum_{j=1}^{n} \alpha_j y_j G(x_j, x) + b, \tag{4}$$

onde $(\alpha_1, ..., \alpha_n.b)$ são os parâmetros estimados do SVM, e $G(x_j, x)$ é o *kernel* utilizado. Neste trabalho o *kernel* é linear, ou seja, $G(x_j, x) = x_j^{'}x$. O *kernel* linear apresenta um bom desempenho com a mínima quantidade de parâmetros de entrada.

5.4. Resultados da Classificação de Ataques

Os resultados obtidos de cada algoritmo supervisionado implementado são apresentados através de duas métricas: a acurácia e a matriz de confusão. A acurácia define o percentual de acerto de classificação no conjunto de dados de treino. A matriz de confusão especifica de forma clara o número de falsos positivos de cada classe. Essa matriz é uma matriz no formato Classe Real versus Classe Prevista. As linhas representam os elementos que pertencem à classe de verdade, e as colunas os elementos que foram atribuídos à classe pelo algoritmo de classificação. Portanto, os elementos na diagonal dessa matriz representam os acertos de classificação, já que os elementos pertencem à classe e foram classificados como pertencentes da mesma classe. Assim, qualquer elemento fora da diagonal representa um erro. Essa métrica foi escolhida, porque a partir dela é possível se determinar todas as outras medidas, como falso positivo e taxa de acerto de cada classe.

Os resultados mostram uma alta acurácia na classificação de ataques em cada um dos algoritmos utilizados. A Tabela 1 mostra a comparação da acurácia dos distintos algoritmos. Na tabela é possível observar que os três algoritmos utilizados no sistema apresentam uma acurácia semelhante, perto do 95%.

Tabela 1: Comparação de Acurácia dos Distintos Algoritmos Utilizados na Classificação de Ataques.

	Árvore de Decisão	Rede Neural	Máquina de Vetores de Suporte
Acurácia	95.9984%	95.1271%	95.8103%

As Tabelas 2, 3 e 4 mostram as matrizes de confusão dos algoritmos aplicados na classificação de ataques. Na diagonal principal da matriz de confusão observa-se os acertos em cada uma das classes. Assim, é analisada com maior profundidade a acurácia de cada classificador mostrada na Tabela 1. Tanto a Tabela 2 como a Tabela 4 mostram o melhor desempenho, já que elas apresentam um menor número de Falsos Positivos (FP).

Tabela 2: Matriz de Confusão da Árvore de Decisão

	Normal	DoS	Varredura
Normal	29126	1	0
DoS	60	5845	0
Varredura	8	1782	9434

Tabela 3: Matriz de Confusão da Rede Neural

	Normal	DoS	Varredura
Normal	28807	253	67
DoS	95	5810	0
Varredura	1839	0	9385

Tabela 4: Matriz de Confusão da Máquina de Vetores de Suporte

	Normal	DoS	Varredura
Normal	29087	0	40
DoS	63	5842	0
Varredura	1835	0	9389

5.5. Detecção de Anomalias pela Distribuição Gaussiana

Ataques novos (*zero-day attacks*) são muito difíceis de serem detectados, pois não há dados ainda sobre como são realizados. Por isso, a proteção contra ataques desconhecidos é essencial para um maior nível da segurança de redes de computadores. A detecção de anomalias consegue assim descobrir ataques novos.

Neste artigo, a detecção de anomalias é proposta pelo distanciamento de uma distribuição gaussiana. Logo, as anomalias são detectadas a partir da média e da variância de cada característica do conjunto de dados de treino com amostras normais. Assim, as anomalias ocorrem quando é detectada uma distância para média maior do que o valor de um limiar vezes a variância em pelo menos uma das características. São analisadas as 8 características obtidas pela transformação linear da PCA.

A implementação em tempo real requer a detecção de anomalia do fluxo de dados à medida que os dados vão chegando no sistema. A anomalia é detectada se pelo menos uma das seguintes desigualdades for verdadeira para qualquer característica j, levando em conta as médias μ_j e as variâncias σ_j^2 calculadas no treino:

$$X_j > \mu_j + limiar * \sigma_j^2 \qquad (5) \qquad\qquad X_j < \mu_j - limiar * \sigma_j^2 \qquad (6)$$

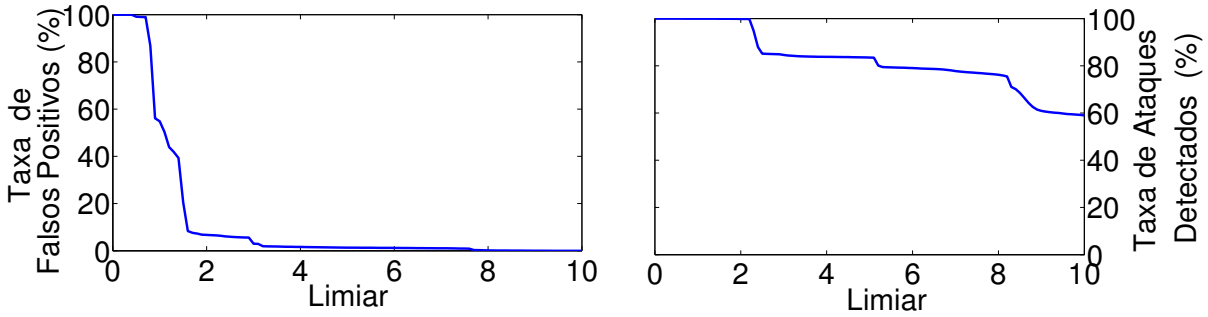
O sistema proposto permite o treino em tempo real devido à arquitetura lambda. Assim, o algoritmo torna-se adaptativo, o que é fundamental para a detecção de anomalias, pois o comportamento da rede pode mudar com o tempo. Logo, quando um fluxo novo chega ao sistema e ele não é detectado como anomalia pelas desigualdades (5) e (6), os parâmetros μ_j e σ_j^2 de cada característica são adaptados, considerando esse novo fluxo. Os parâmetros de uma distribuição normal são calculados da seguinte forma:

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} X_j \qquad (7) \qquad\qquad \sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(X_j - \mu_j)^2 \qquad (8)$$

Portanto, os valores correntes dos somatórios e da quantidade de amostras N são armazenados no sistema e incrementados quando um novo fluxo chega ao sistema, considerando cada característica X_j do fluxo. Assim, os parâmetros da gaussiana são sempre atualizados de acordo com os fluxos considerados benignos, garantindo adaptabilidade na detecção de anomalia.

Os resultados são mostrados na Figura 5 para diferentes valores da variável $limiar$. Os

resultados de taxa de falsos positivos foram obtidos ao se avaliar o conjunto de dados *dataset* de treino normal e os de taxa de detecção de ataque com todos os ataques que estão no conjunto de dados. Ao se escolher um valor baixo de limiar, quase todos os ataques são detectados, porém o número de falsos positivos também é maior. Já um valor elevado resulta em menos falsos positivos, mas também uma taxa de detecção menor. Com $limiar$ igual a 2, a taxa de falsos positivos é de 6,7% e a de detecção de 99,8%. Com $limiar$ igual a 3, essas taxas são de 3,0% e de 84,5%. Esse parâmetro então deve ser escolhido de acordo com a aplicação, levando em conta seus requisitos de segurança e o quanto de falsos positivos é aceitável.



(a) Taxa de Falsos Positivos. **(b) Taxa de Detecção de Ataques.**

Figura 5: Taxa a) de falsos positivos e b) de detecção de ataques em função do limiar em número de variância. Quanto menor o limiar, mais ataques são detectados, porém maior é a taxa de falsos positivos.

5.6. Desempenho em Tempo Real

O tempo de detecção de uma ameaça é crucial para manter a segurança em redes. Se a detecção demorar demais, não há qualquer reação que possa ser tomada para neutralizar a ameaça. Através do processamento de fluxos e da arquitetura lambda, é possível usar os parâmetros de um treino prévio para a detecção em tempo real. O protótipo do sistema foi implementado num ambiente com 4 máquinas virtuais para testar o desempenho em tempo real. As máquinas virtuais executavam sobre uma máquina com processador Intel Core i7-2600 e com 16 GB de memória RAM. A medida de desempenho foi feita com a implementação da rede neural, pois ela apresenta a maior complexidade computacional entre os algoritmos implementados. O sistema foi capaz de processar em média 3,57 milhões de fluxos por minuto com incerteza de 156 mil, medidas com intervalo de confiança de 95%. Portanto, o tempo de detecção de cada ataque é cerca de 17 microssegundos o que permite tentativas de defesas eficazes diminuindo o risco do ataque.

6. Conclusão

Este artigo propõe um sistema de detecção de ameaças em tempo real por processamento de fluxos. O sistema proposto usa a arquitetura lambda, que combina os paradigmas de processamento em fluxos e em lotes. O processamento em lotes, em tempo diferenciado, permite uma adaptação automática dos algoritmos de aprendizado de máquinas para serem usados em tempo real. Para avaliar o desempenho do sistema proposto, um conjunto de dados baseado em tráfego real de rede foi criado para a avaliação da proposta. Os pacotes capturados foram separados em fluxos rotulados através de 24 características. Para aumentar a eficiência do sistema a dimensionalidade do conjunto de dados foi reduzida através do algoritmo de análise de componentes principais. Foram utilizados os algoritmos de aprendizado de máquinas árvore de decisão, rede neural e máquina de vetores de suporte para a classificação em tempo real dos ataques. Os

resultados mostram uma acurada classificação de ataques dos três algoritmos com valores maiores que 95%, sendo o algoritmo de árvore de decisão foi o de melhor acurácia com 95,99%. Os algoritmos implementados também apresentaram uma baixa taxa de falsos positivos. Além disso, foi implementado um algoritmo de detecção de anomalias para detectar ataques desconhecidos que apresentou um bom compromisso entre as taxas de falsos positivos e detecção de ataques de acordo com o limiar escolhido. Por exemplo, pode-se obter uma taxa de detecção de ataques de 99,8% com 6,7% de taxa de falso positivo ou, com um limiar diferente, uma taxa de detecção de 84,5% e apenas 3,0% de falsos positivos. O processo de detecção de anomalias é adaptativo, já que os parâmetros são atualizados em tempo real. O protótipo implementado utiliza ferramentas de processamento de fluxos, como o Apache `Storm` e o Apache `Kafka` e, por isso, consegue um baixo tempo na detecção das ameaças, possibilitando estratégias de defesa. O sistema apresenta um bom desempenho em tempo real analisando até 3,57 milhões de fluxos por minuto.

A implementação do sistema proposto como serviço em nuvem será realizada em um trabalho futuro. Esse serviço deve apresentar uma característica elástica no fornecimento dos recursos. O desenvolvimento de detecção de outros tipos de ataques, como ataques na camada de aplicação também será um trabalho futuro.

Referências

Amini, M., Jalili, R. e Shahriari, H. R. (2006). RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. *Computers & Security*, 25(6):459 – 468.

Andreoni Lopez, M., Figueiredo, U. d. R., Lobato, A. G. P. e Duarte, O. C. M. B. (2014). BroFlow: Um sistema eficiente de detecção e prevenção de intrusão em redes definidas por software. *XII WPerformance (XXXIV CSBC)*, páginas 1919–1932.

Buczak, A. e Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, (99):1–26.

Clay, P. (2015). A modern threat response framework. *Network Security*, 2015(4):5–10.

Costa, L. H. M. K., de Amorim, M. D., Campista, M. E. M., Rubinstein, M., Florissi, P. e Duarte, O. C. M. B. (2012). Grandes massas de dados na nuvem: Desafios e técnicas para inovação. Em *SBRC 2012 - Minicursos*.

Du, Y., Liu, J., Liu, F. e Chen, L. (2014). A real-time anomalies detection system based on streaming technology. Em *Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 2, páginas 275–279. IEEE.

Holtz, M. D., David, B. M. e de Sousa Júnior, R. T. (2011). Building scalable distributed intrusion detection systems based on the mapreduce framework. *Revista Telecommun*, 13(2).

Lee, W., Stolfo, S. J. e Mok, K. W. (1999). Mining in a data-flow environment: Experience in network intrusion detection. Em *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 114–124. ACM.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., Weber, D., Webster, S. E., Wyschogrod, D., Cunningham, R. K. et al. (2000). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. Em *Proceedings of DARPA Information Survivability Conference and Exposition. DISCEX'00.*, volume 2, páginas 12–26. IEEE.

Liu, Q., Lui, J., He, C., Pan, L., Fan, W. e Shi, Y. (2014). SAND: A fault-tolerant streaming architecture for network traffic analytics. Em *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, páginas 80–87. IEEE.

Lobato, A. G. P., da Rocha Figueiredo, U., Andreoni Lopez, M. e Duarte, O. C. M. B. (2014). Uma arquitetura elástica para prevenção de intrusao em redes virtuais usando redes definidas por software. *Anais do XXXII SBRC 2014*, páginas 427–440.

Marz, N. e Warren, J. (2013). *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications Co., Greenwich, CT, USA, 1st edition.

Ponemon, I. e IBM (2015). 2015 cost of data breach study: Global analysis. `www.ibm.com/security/data-breach/`. Acessado: 27/12/2015.

Porto, F. e Ziviani, A. (2014). Ciência de dados. Em *3o. Seminário de Grandes Desafios da Computação no Brasil*. SBC.

Rai, K. e Devi, M. S. (2013). Intrusion detection systems: A review. *Journal of Network and Information Security Volume*, 1(2).

Ringberg, H., Soule, A. e Rexford, J. (2008). Webclass: adding rigor to manual labeling of traffic anomalies. *ACM SIGCOMM Computer Communication Review*, 38(1):35–38.

Rychly, M., Koda, P. e Smrz, P. (2014). Scheduling decisions in stream processing on heterogeneous clusters. Em *Eighth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, páginas 614–619.

Sangkatsanee, P., Wattanapongsakorn, N. e Charnsripinyo, C. (2011). Practical real-time intrusion detection using machine learning approaches. *Computer Communications*, 34(18):2227 – 2235.

Solaimani, M., Khan, L. e Thuraisingham, B. (2014). Real-time anomaly detection over VMware performance data using storm. Em *IEEE 15th International Conference on Information Reuse and Integration (IRI)*, páginas 458–465. IEEE.

Sommer, R. e Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. Em *IEEE Symposium on Security and Privacy (SP)*, páginas 305–316. IEEE.

Sperotto, A., Sadre, R., Van Vliet, F. e Pras, A. (2009). A labeled data set for flow-based intrusion detection. Em *IP Operations and Management*, páginas 39–50. Springer.

Stroeh, K., Madeira, E. R. M. e Goldenstein, S. K. (2013). An approach to the correlation of security events based on machine learning techniques. *Journal of Internet Services and Applications*, 4(1):1–16.

Tavallaee, M., Bagheri, E., Lu, W. e Ghorbani, A.-A. (2009). A detailed analysis of the KDD CUP 99 data set. Em *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*. IEEE.

Wu, K., Zhang, K., Fan, W., Edwards, A. e Yu, P. S. (2014). RS-Forest: A rapid density estimator for streaming anomaly detection. Em *IEEE International Conference on Data Mining (ICDM)*, páginas 600–609.

Zhao, S., Chandrashekar, M., Lee, Y. e Medhi, D. (2015). Real-time network anomaly detection system using machine learning. Em *11th International Conference on the Design of Reliable Communication Networks (DRCN)*, páginas 267–270. IEEE.

A Collaboration Model to Recommend Network Security Alerts Based on the Mixed Hybrid Approach

Arthur de Moura Del Esposte[1], Rodrigo Campiolo[1,2], Fabio Kon[1], Daniel Batista[1]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (IME-USP)
Rua do Matão, 1010 – 05508-090 – Cidade Universitária – São Paulo – SP – Brasil

[2]Universidade Tecnológica Federal do Paraná (UTFPR) - Campo Mourão - PR - Brasil

{esposte, kon, batista}@ime.usp.br, rcampiolo@utfpr.edu.br

Abstract. *A high number of cyber security alerts are shared every second in different medias like forums, mail lists, and online social networks. The flood of alerts complicates the network administrator's job, since not all cyber security alerts are important for his/her specific environment. Thus, recommender system techniques could be properly used to filter cyber security alerts based on network administrator ratings and preferences. This paper presents a collaboration model to recommend cyber security alerts for network administrators, helping them to focus on the relevant alerts. To evaluate the model, an offline experiment is executed. Partial results showed that our model can be used to recommend cyber security alerts.*

1. Introduction

The Internet has grown significantly in recent years, especially in number of users, offered services, and transferred data. Thereby, most human activity fields, such as education, health, business, economy, and social interactions have evolved with the expansion of the Internet. More and more people, enterprises, governments, and critical infrastructures depend on information and communication technologies (ICT) and the infrastructure that make up cyberspace.

To address the challenges related to ICT dependence, such as the confidentiality, availability and integrity, it is necessary to increase the cyber security in all layers related to it, from physical components to organizational procedures. However, the risk to information and computer assets comes from a broad spectrum of threats, that grows as new vulnerabilities emerge [GCHQ and Cert-UK 2015a].

Formally, a vulnerability can be defined as an instance of failure in the specification, development or software configuration so that its execution can violate security polices, implicitly or explicitly [Krsul 1998]. Vulnerabilities can be maliciously exploited to allow unauthorized access, privileges changes and denial of service. Although the ICT is composed of many physical, logical or organizational components, most of the exploitable vulnerabilities is present in software. According to the Internet Security Threat Report [GCHQ and Cert-UK 2015b], in 2014 76% of websites contained vulnerabilities.

The more new cyber security issues arise, the more network administrators must update and improve their knowledge about these issues to implement preventive actions on computer networks and services. As long as many new attacks occur and new vulnerabilities are discovered, many alerts and news are released in all sorts of media. Specialized

Websites and email lists are examples of traditional medias used to warn about security threats, though they are not always effective in quickly publishing recent threats [Frei et al. 2006]. As discussed by Santos and collaborators [Santos et al. 2013], alternative decentralized medias, such as social networks, provide rich and heterogeneous data sources that can be used as means of spreading security notifications. This speech is reinforced by Surjandari and colleagues [Surjandari et al. 2015] who claim that social networks are valuable platforms for tracking and analyzing information, since people post real time messages about their opinions on a variety of topics and disseminate information quickly and collaboratively.

Despite the large number of cyber security news and warnings generated over the Internet every day, a network administrator not always gets relevant information. The large number of data sources can become a problem since it is a hard task to find or filter relevant information of interest. Historically, a natural way to accomplish this is relying on recommendations from people of similar interest profile or on expert advice. Ekstrand et al. [Ekstrand et al. 2011] expose that computer-based systems provide the opportunity to expand the set of people from whom users can obtain recommendations. They also enable us to mine users' history and stated preferences for patterns that neither them nor their acquaintances identify, potentially providing a more finely-tuned selection experience. The set of computer systems that provide those services are called *recommender systems*.

In view of what has been mentioned, this paper focuses on the design and evaluation of a recommender model for cyber security alerts extracted from external unstructured data, based on a *mixed hybrid* approach, to properly improve the hard task to get valuable cyber security alerts and news. Two main recommendation techniques are used in the designed model: Content-based and Collaborative Filtering. We also present a prototype implementation of this model, which is used to evaluate it through an offline experiment.

The remainder of this paper is structured as follows: Section 2 presents the main concepts related to recommender systems. In Section 3 we review and discuss related works. Section 4 describes the design and how we defined the recommender model, followed by the explanation of the method used to evaluate it. We show the experimental results in Section 5. In Section 6, we present the conclusion and future directions.

2. Theoretical Background

This section summarizes the main concepts related to this work. First, we describe the Top-N recommendation problem. After, we describe what are recommender systems and the two categories of algorithms that were used to design the hybrid recommender model presented in Section 4.

2.1. The Top-N Problem

The problem we aim to solve is to recommend a set of news and alerts that can be useful for a given network administrator. This problem is characterized as the Top-N recommendation problem which aims to identify a set of N items that will be of interest to certain user [Karypis 2001]. To avoid problems with efficiency or accuracy, N should be chosen carefully. If N is too large, an excessive amount of memory will be required to store the neighborhood lists and predicting ratings will be slow. On the other hand, selecting a too

small value for N may reduce the coverage of the recommender method, which causes some items to be never recommended [Ricci et al. 2011].

2.2. Recommender Systems

A recommender system, as defined by Ricci and colleagues in their book *Recommender Systems Handbook* [Ricci et al. 2011], is a set of tools and techniques that provide suggestions for items of interest to a user. The problem of recommendation can be seen as the problem of estimating the user rating for items not yet rated by the user. Items predicted with high rating for a given user can be offered by the system as a recommendation [Glauber et al. 2013].

In this work, we are interested in the two main approaches for recommendation: Collaborative Filtering and Content-based Filtering. Hereafter, we discuss both approaches and related problems in more detail. Last, we describe hybrid recommender approaches, highlighting the mixed approach that is used in our proposed model.

Collaborative Filtering: recommends to a given user the items that other users with similar tastes liked in the past. The similarity in taste of two users is calculated based on the similarity in the rating history of the users [Feldman and Sanger 2007]. This approach is the most popular and widely implemented technique in recommender systems [Ricci et al. 2011]. An important point of this approach is that the recommendations are based on the quality of items as evaluated by peers. However, Collaborative Filtering suffers from the *cold-start* problem of handling new items or new users in the system, once there is no ratings related to neither new items or new users. Another consequent difficulty with this approach is the sparsity of the ratings space, meaning that most entries are unknown.

Content-based Filtering: the general principle of this approach is to identify the common characteristics of items that have received a favorable rating from a user u, and then recommend to u new items that share these characteristics [Feldman and Sanger 2007]. The Content-based approach suffers of limited content analysis if the system has only a limited amount of information on its users or the content of its items. This approach does not consider the quality of the items. Another important issue related to the Content-based approach is the problem to recommend items that are different but still interesting to the user.

Hybrid Approach: combines two or more recommender filtering to provide more sophisticated and complete recommendations. Burke [Burke 2007] explores seven strategies for hybrid recommendations, which were identified in his earlier survey of hybrids [Burke 2002], where he identifies possible combinations of recommendation approaches. We are mainly interested in the *mixed hybrid* approach, which presents recommendations of its different components side-by-side in a combined list. The main challenge in this type of recommender is to choose how to rank the recommended items from different approaches.

[Ricci et al. 2011] defines three main elements to be specified in a recommender model:

Items: item is an entity that is recommended in a recommender system. An item has the main attributes, in which a user is interested, and other structures that are used to

rate or value them. In our problem, an item is defined as a security alert and its content elements are the main attributes.

Users: a user may have very diverse goals and characteristics whose information are used by recommendation algorithms to make new item recommendations. In our domain, the users are network administrators who are technically interested in cyber security.

Transactions: a transaction refers to a potential interaction between a user and the recommender system. The set of transactions define how items can be rated. All recommender algorithms use data collected from some transactions to perform a new recommendation.

3. Related Works

Several works have researched methods to obtain relevant security alerts to support the prevention of cyber attacks. [Apel et al. 2009] and [Flegel et al. 2010] suggested collaborative approaches based on the exchange of information among organizations to prevent and detect security threats in advance. [Grobauer et al. 2006] proposed an organizational framework to share, correlate, and analyze cyber security alerts extracted from multiple organizations. Unlike those three previous works, our research aims to evaluate a model that can be directly used by network administrators, instead of only support organizational environments.

[Bonchi et al. 2011] and [Surjandari et al. 2015] identified business opportunities in the rich content generated through collaboration between users in social networks. Santos et al. [Santos et al. 2013] also explored the potential for collaboration in social networks to extract cyber security alerts from *Twitter* that can lead network administrators to apply security policies faster. Even though Santos et al. have used heuristic techniques and unsupervised learning to properly filter cyber security messages, they evidenced the difficulty of dealing with the semantics of those messages. By applying recommender systems techniques, our work continues exploring the collaboration phenomena between users and the progress made by [Santos et al. 2013].

[Karypis 2001] discussed and compared some algorithms to solve the problem of top-*N* recommendations by conducting several offline experiments. [Ekstrand et al. 2011] and [Font et al. 2013] used and evaluated several different recommender approaches, whereas [Sarwar et al. 2001] and [Zheng and Li 2010] adapted traditional recommender methods based on collaborative filtering and content-based approaches to different practical applications. [Burke 2002] exposed some hybrid approaches to build recommender systems using more than one technique, while [Glauber et al. 2013] explored the *mixed* hybrid approach for given names, similar to the hybrid approach we adopt in our recommender model.

Our work advances the state of the art in cyber security by proposing a new model for gathering relevant information on cyber security alerts based on recommender system methods. Thus, the proposed recommender model aims to address the open issues arising from the progress made in [Santos et al. 2013] by applying collaborative techniques to properly classify cyber security alerts.

4. Recommender Model Design

This section presents the proposed recommender model for cyber security alerts and the methodology used for its design. We followed the process proposed by Ricci et al. [Ricci et al. 2011], which addresses the recommendation problem in three dimensions:

- Users: who are the users, what are their goals?
- Data: what are the characteristics of the data on which recommendations are based?
- Application: what is the application the recommender is part of?

The following subsections describe how we adopt this process. First, Section 4.1 describes how we requested and analyzed the requirements for the recommender model. After gathering enough information, we defined the data model specification, selected the proper recommender techniques and composed the mixed hybrid solution, as detailed in Section 4.2. Finally, Section 4.3 exposes the design of the running offline experiment.

4.1. Requirements Inception and Analysis

The main objective of the present work is to build a recommender model to recommend cyber security alerts, collected from external data sources. The recommender model is aimed at general users with network administrator profile, technically interested in cyber security. We intend to ensure that the model always recommend some items even if the user has never informed their preferences and even at the first interaction of the user with a system implementing the model.

Following the main definitions of entities related to a recommender system [Ricci et al. 2011], the first step to design the recommendation model is to identify some initial information about the users who will compose the model, understanding their key identifying characteristics, their skill levels, and their prior experience with similar systems. It is also important to identify the main characteristics about cyber security alerts which will match the model items and the possibilities of interactions between users and items.

For this purpose, we applied a qualitative survey for dozens of Brazilian network administrators professionals, getting 20 responses, through ESurv[1], an online web-based free tool for create surveys. This survey was designed with four parts:

- *Profile Survey*: to identify some general characteristics of potential users.
- *Cyber Security Survey*: to identify how much the users know about cyber security issues and whether they use any communication channel to be up to date about those issues.
- *Interest Assessment*: to identify requirements that must be considered into the recommender model conception. It also helps to get what are the main elements that must compose the recommender model item by asking the users what are the most important elements of a cyber security alert. Finally, this topic also provides questions that will be used to establish some interactions between users and items.
- *Use Cases*: this section was designed to simulate some use cases of an hypothetical recommender system to evidence in more detail potential users' interactions.

The survey is available online[2] as well as the results[3].

[1]http://esurv.org/ - Accessed at December 22, 2015

[2]http://bit.ly/1Rk0nR8 - Accessed at December 22, 2015

[3]https://gitlab.com/konsilo/survey/blob/master/results.pdf - Accessed at December 22, 2015

4.2. Model Specification

To build the proposed recommender model, we need to properly define the User and the Item elements. Both elements can be defined as a set of data structures that represent the real world. For both definitions, we used the applied survey described in Section 4.1.

About the User data structure, on average, the respondent network administrators have about 8 years of work experience. Most of them have a good knowledge about cyber security and often read cyber security news. However, only half of them interact with other network administrators, mainly through mailing lists and chat rooms.

A positive result we had in the survey is that all the 20 respondents are interested in using a collaboration system to get and share cyber security alerts and news. However, most of them would never provide some information as input for a recommender system like *personal name* or the *institution they work on*, as can be seen in Figure 1, which shows the percentages of respondents that selected each information they would never provide.

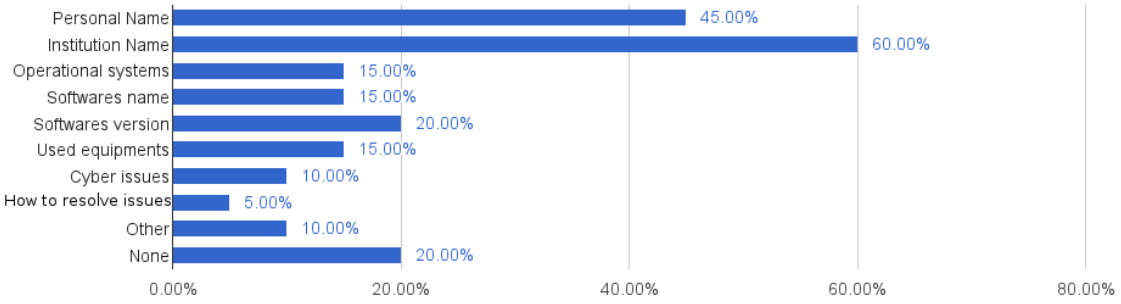


Figure 1. Information that users would never provide

Through the survey results it was also possible to notice that some types of information can be used to determine user preferences, such as operating system and type of attack related to an alert.

The recommender model items are the cyber security alerts, which are mostly available as unstructured data extracted from heterogeneous data sources. As evidenced in the survey results, the following attributes are the most important to be included into our item data structure, from the most important to the less important: (1) title, (2) information source, (3) level of critically, (4) related software/hardware, (5) keywords (tag).

Other collected data from the applied survey demonstrated how users would rate some items, as shown in Figure 2.

As can be seen in Figure 2, the most popular rating mechanism is to mark an alert as urgent or critical. Another popular rating mechanism is vote, which can be positive or negative and can be influenced by different motivations. Open comments also appears as a popular feedback input but we did not add any transaction related to it at first, mainly because it provides more subjective than objective data, tough it could be used in a real implementation of the proposed model. We also added the Categorize rating mechanism

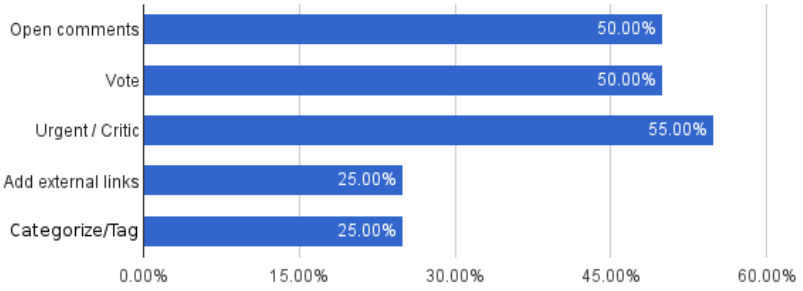


Figure 2. Rating mechanisms

to our recommender model because it can be useful to support the application of content-based filter approach, provided that it is used to define cyber security alerts' features. Therefore, the transactions of the proposed recommender model are shown in Figure 3.

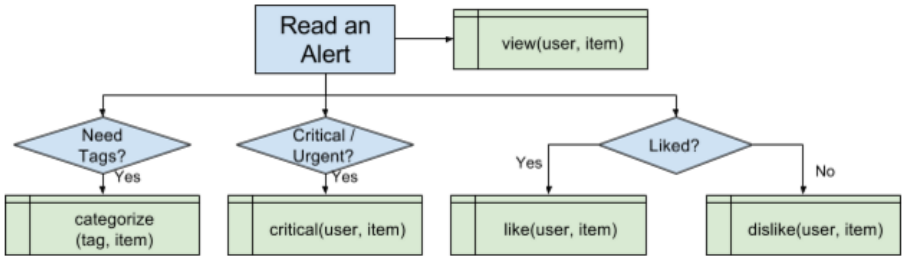


Figure 3. Recommender model transactions

Figure 3 shows the five transactions of our recommender model. They are described below:

- **view**: registers an item visualization by an user. In our model, it must be used for two purposes:
  - Estimate the most popular alerts so that it can be used in general recommendations of cyber security alerts.
  - Whenever a given user reads a security alert, the alert should not be recommended for this user again. Thus, the *view* transaction is important to remove read alerts of new recommendations.
- **categorize**: registers a new category to an alert. As many cyber security alerts are extracted from unstructured data sources, we introduced the *categorize* transaction to supply enough information about cyber security alerts in order to support the Content-based approach. For this purpose, collaborative tagging has emerged as a plausible alternative, despite the related challenges such as tag scarcity or ambiguous labelling [Font et al. 2013]. Alerts categorization can also be supported by text mining techniques by retrieving useful information from alerts that can correctly categorize an item.
- **critical**: registers a user opinion about the urgency of an alert. Although it was the most popular rating mechanism in our survey, it is only considered for general

ranking. On the other hand, it must have a higher weight than *views* and *likes* transactions for general ranking purposes.

- **like**: registers a positive relation between an user and an item. This can be interpreted as if the user showed interest or approval of the rated cyber security alert. It is important for the following purposes:
  - Estimate the most liked alerts so that it can be used in general recommendations of cyber security alerts.
  - Recommend new cyber security alerts using Collaborative Filtering, since it can be used to infer the similarity between users.
  - Recommend new cyber security alerts using Content-based approach, since it defines the main interests of an user through alerts' categories.
- **dislike**: registers a negative relation between a user and an item. This can be interpreted as if the user did not approve the rated cyber security alert content. In our model, it is important to find false or incorrect alerts to remove them. Therefore, the *dislike* transaction is neither used for the Collaborative Filtering nor the Content-based approaches, but it is useful to clean bad items from the database.

Figure 4 shows the recommender model flow, displaying the user's (*blue*) and system's (*red*) actions and how the different recommender approaches are combined to recommend cyber security alerts for network administrators.

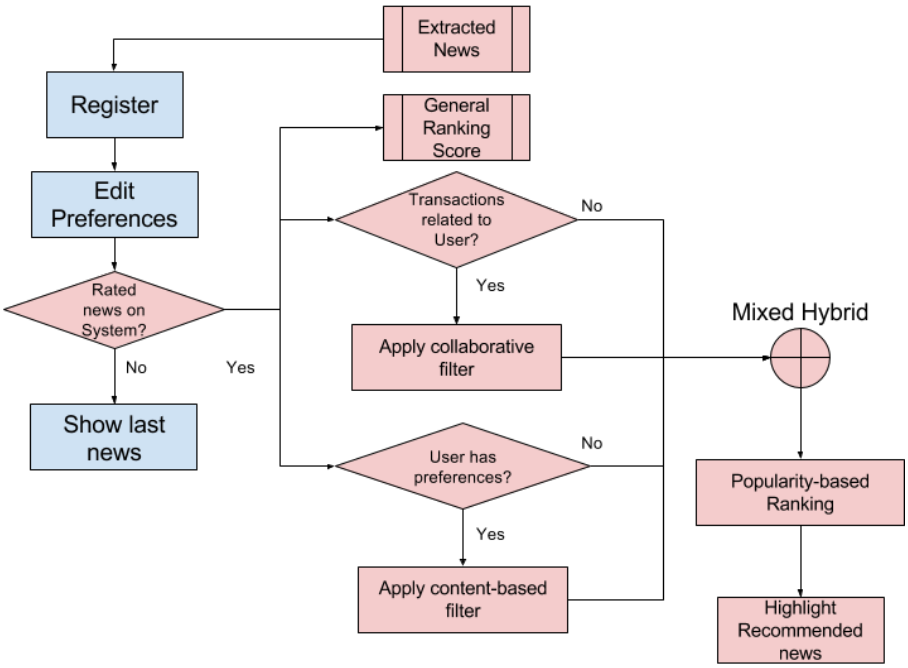


Figure 4. Recommender model main flow

As can be observed in Figure 4, since we have rated items, we must perform three computations separately, which may be done periodically: compute the general ranking score of all items, compute the collaborative filtering and the content-based approaches' recommendations. We compute the ranking score based on the Bayesian Average, as defined in Equation 1, which incorporates existing information related to our data set to minimize the impact of large deviations. The ranking score aims to define a general score

to rank items, since it only considers items and ignore any data of specific users. The greater is the $ranking\_score(i)$, the better ranked an item *i* will be.

$$ranking\_score(i) = \frac{(\bar{v} \cdot \bar{r}) + (v_i \cdot r_i)}{\bar{v} + |r_i|} \tag{1}$$

where $\bar{v}$ is the average number of *votes* (likes, dislikes or critical) across all alerts; $\bar{r}$ is the average rating of all alerts; v_i is the total votes of item *i*; and r_i is the rating of the item *i*, which is defined in the Equation 2.

$$r_i = \alpha c_i + \beta l_i - \phi d_i \tag{2}$$

Equation 2 computes the rating score of an item *i* adding up the positive votes and subtracting the negative votes, where c_i is the total number of votes that classify *i* as *critical*, l_i is the total number of *likes*, and d_i is the total number of *dislikes*. It is also important to give different weights for the elements of Equation 2, as denoted by α, β and ϕ, which may vary according to the implementation of the model. Consistent with our survey results described earlier in this Subsection, we suggest the following weights showed in Equation 3, where *critical* votes have the greatest weight, followed by *dislikes* which is weighted twice larger than *likes*, as we want to remove bad alerts.

$$r_i = 4c_i + l_i - 2d_i \tag{3}$$

When there is no rated item, for instance at the initial state of the system, the model shows cyber security alerts ranked by the most recent. Otherwise, we apply the Content-based and Collaborative Filtering separately to generate two lists of recommended items.

To generate the first list with the top-*N* recommendations, we apply the Collaborative Filtering user-based approach, which is the most successful technology for building recommender systems, and is extensively used in many commercial recommender systems [Karypis 2001]. This approach aims to determine the similarity between users and recommended items to a given user that his/her fellows liked. Consequently, items frequently liked by the various members of the group can be used to form the basis of the recommended items. Thus, it is necessary to perform two steps to compute the list of top-N recommendations for a given user *u*.

First, we compute the similarity between users to find the *neighbors*, the set of users that have a history of agreeing with the target user *u*. To compute the similarity value of two users *u* and *v*, we use the Jaccard similarity coefficient defined in Equation 4, which is suitable for binary votes.

$$J(u, v) = \frac{|\ U_l \cap V_l\ |}{|\ U_l \cup V_l\ |} = \frac{|\ U_l \cap V_l\ |}{|\ U_l\ | + |\ V_l\ | - |\ U_l \cap V_l\ |} \tag{4}$$

where U_l is the set of items liked by user *u* and V_l is the set of items liked by user *v*. The $J(u, v)$ can take values between 0 and 1.

Equation 4 is defined as the ratio between the amount of items liked by both two users, and the size of the union of the items liked by each user. The greater is the Jaccard

similarity coefficient for two users *u* and *v*, the greater the similarity of these two users.

Once the set of *k neighbors* of user *u*, denoted as $Neighbor(u)$, is formed, the last step is to combine the preferences of neighbors to recommend the top-*N* items for the target user *u*. For this purpose, we measure whether user *u* may like an item *r* by computing Equation 5, as Zheng & Li have done in a previous work [Zheng and Li 2010]. Since we are interested in recommending new items for a target user *u*, we do not compute Equation 5 for items that had already been viewed by *u*.

$$score(u, r) = \frac{\sum_{v \in Neighbor_{(u)}} R_{v,r} \times J(u, v)}{\mid \sum_{v \in Neighbor_{(u)}} J(u, v) \mid} \tag{5}$$

where $R_{v,r}$ is the rate of user v over the item r.

In our model, the second list is generated independently with the Content-based approach through the match of weighted tags represented by the user's interests and item's categories. Whenever a user likes an item, each one of that item's categories tags is added to the user's interests and its weight is set to 1, if it is a new tag, or incremented by 1 otherwise.

Equation 6 is an adaptation of the method presented by Zheng and Li [Zheng and Li 2010], as we added weight to tags based on user liked items, to compute content-based recommendations. It expresses our tag-weight strategy to measure how much a user *u* is interested in a given item *i*.

$$w_{tag}(u, i) = \frac{\sum\limits_{t_i \in tag(i)} weight(u, t_i)}{\sum\limits_{t_j \in tag(u)} weight(u, t_j)} \tag{6}$$

where *tag(i)* represents the set of tags which categorizes an item *i*; *tag(u)* represents the set of tags of interest of a given user *u*; $weight(u, t_i)$ denotes the weight of the tag t_i for the user u, and while $weight(u, t_j)$ denotes the weight of the tag t_j for the user u.

Equation 6 always takes a real number between 0 and 1, and the higher the weight is, the more interest a user has in the item *i*. Note that unlike our Collaborative Filtering approach, our Content-based method does not depend on other users' information to generate a top-*N* list of recommended items.

After performing the Collaborative filtering and the Content-based algorithms, we are ready to apply the mixed hybrid approach to select the top-*N* items to recommend for a target user. Note in Figure 4 that we also mix the two custom generated lists with the popularity-based ranking defined in Equation 1 to always include items considered important by the user community. Since we want to find the top-*N* items to recommend to target user *u*, we select the top ranked *N* items not viewed by *u* and compute the $mixed\_score$ of the item *i* for the user *u*, by applying Equation 7.

$$mixed\_score(u, i) = 2^{\gamma} \cdot ranking\_score(i) \tag{7}$$

where γ is two if i is recommended in both recommender approaches, one if i is only recommended by one of the approaches, and zero otherwise.

Finally, we can get the final recommendation list of items selecting the N items with highest *mixed_score*.

4.3. Evaluation

As described in [Ricci et al. 2011], to evaluate a recommender model we must identify the set of properties that may influence the success of a recommender system or model in the context of a specific application. Then, we can evaluate how the system performs on these relevant properties. In the proposal model, the *offline experiment* approach was used to evaluate the properties of the model we built.

Since there are no available data sets with the necessary cyber security alerts information, to evaluate the proposed recommender model, we used experimental data from MovieLens[4], a recommender system research website created to recommend movies based on user ratings. Although the set of rated items in this data set is not related to cyber security alerts, we can use it since it has rates and categories related to items. As our model is composed of two different recommender approaches, Collaborative Filtering and Content-based, we evaluate them separately. We started our experiment by first dividing the data set into two groups:

- **Training data**: the set of data to simulate the user behavior in our model by recording historical user data. These data corresponds to 80% of the entire data set and are used for computing new recommendations for users in *testing data*.
- **Testing data**: the set of data that is hidden from the model to be compared with the results. These data corresponds to 20% of the entire data set. Since we hide a user *u* rating for an item *i*, we gather the output of our recommender model for *u* and *i*, which can be compared through the defined metrics.

To compare different configurations of the algorithms, we varied the number of items *N* to be recommended, where $N \in \{3, 5, 10, 15\}$.

We can classify each recommendation as: *true positive - TP* (when an interesting item is recommended); *true negative - TN* (when an uninteresting item is not recommended); *false negative - FN* (when an interesting item is not recommended); *false positive - FP* (when an uninteresting item is recommended) [Cremonesi et al. 2008]. Since both the approaches Collaborative Filtering and Content-based are used to generate a list of *N* recommendations, we evaluate the *precision* and *recall* of them, as explained below:

- **Precision**: measure the ability to recommend *only* what is relevant to the user. The precision is computed as expressed in Equation 8.

$$precision = \frac{TP}{TP + FP} \tag{8}$$

- **Recall**: measure the ability to recommend *everything* that is relevant to the user. The recall is computed as expressed in Equation 9.

$$recall = \frac{TP}{TP + FN} \tag{9}$$

[4]http://grouplens.org/datasets/movielens/ - Accessed at December 22, 2015

Both metrics take values between 0.0 and 1.0, where the higher the value, the better the algorithm's performance in the measured property.

5. Results Analysis

The experiment was lead on a data set with 207 users, 1287 items and a total of 11088 training ratings. To run the experiment, we developed Konsilo, a web application that implements the proposed recommender model to recommend cyber security alerts extracted from social medias and specialized sites. Konsilo's code is available under the *AGPL V3* license in *https://gitlab.com/konsilo/konsilo* .

Figures 5 and 6 show the comparison between the collaborative filtering and content-based approaches. As mentioned before, we evaluate the precision and recall over different configurations of the *top-N* problem.

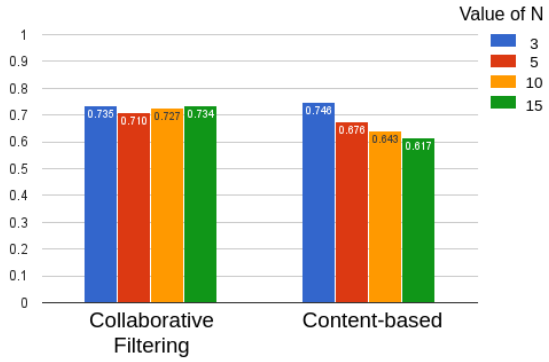


Figure 5. Precision comparison

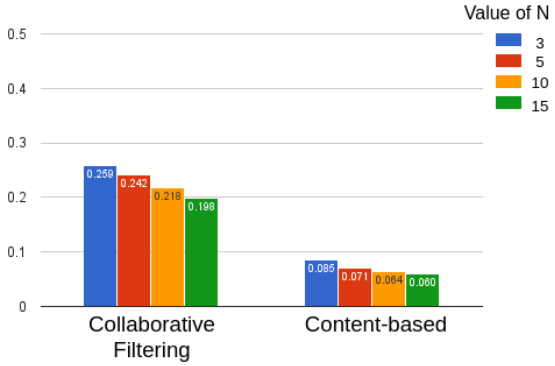


Figure 6. Recall comparison

As can be seen in Figure 5, for different values of *N*, the collaborative filtering had a good precision with little variance. On the other hand, for the content-based approach, the larger the value of *N*, the lower the precision. The best performance was obtained with N equal to 3, but it is important to notice that there is not a large drop in precision for higher values of *N*.

Figure 6 shows the comparison of the recall value between the two recommender approaches. We also measure the recall over different values of *N*. For all configurations, both approaches had low performance related to recall. This may be due the large amount of available items that could be recommended in our experiment, since similar results were found in other *top-N* recommendation experiments [Cremonesi et al. 2008] [Herlocker et al. 2004]. However, it is observable the superiority of collaborative filtering over content-based approach. It is also important to notice that the recall value decreases when increasing the value of *N*.

6. Conclusion and Future Directions

This work advances the state of the art in cyber security by proposing a new model for gathering relevant information on cyber security alerts. To the best of our knowledge, there are no similar works proposed on the literature. This work also shows that recommender system methods and techniques can be explored in the cyber security field to improve access to relevant information for network administrators.

The proposed recommender model is mainly based on two recommender approaches which generate separated lists of recommendations that can be mixed to produce better results and reduce the common problems inherent in them. The model also includes elements of ranking score and other features considered important for network administrators.

It is important to notice that the proposed model is not coupled to technologies, so it can be implemented by any system focused on cyber security alerts. In this work, we developed the web application Konsilo as the first implementation case of the proposed recommender model. Furthermore, Konsilo was used to support the offline experiment to evaluate the proposed collaborative filtering and content-based recommender approaches. In general terms, both approaches demonstrated satisfactory precision for the *top-N* recommendation problem, but poor performance to recall all the interesting items for users.

Future work can be performed to overcome the existing limitations in the present work. We plan to use the GT-EWS[5] project to enable the recommendation of alerts and filtering false positives. We also want to release Konsilo in a production environment as a technological contribution, which also will allow us to carry out new experiments over the recommender model through the behavior of actual users and better evaluate the proposed model as a whole. Both Konsilo and GT-EWS can be used to gather enough information about cyber security alerts and their ratings so that it can be used to build a new data set for future works.

7. Acknowledgments

We thank Fundacão Araucária, Secretaria do Estado da Ciência, Tecnologia e Ensino Superior (SETI-PR), Governo do Estado do Paraná and Rede Nacional de Pesquisa (RNP) for supporting this research and our participation in the event.

References

Apel, M., Biskup, J., Flegel, U., and Meier, M. (2009). Towards early warning systems - challenges, technologies and architecture. In *InCRITIS*, pages 151–164.

Bonchi, F., Castillo, C., Gionis, A., and Jaimes, A. (2011). Social network analysis and mining for business applications. 2:22:1–22:37.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. In *UMUAI 12*, pages 331–370.

Burke, R. (2007). Hybrid web recommender systems. In *The Adaptive Web*, pages 377–408.

Cremonesi, P., Turrin, R., Lentini, E., and Matteucci, M. (2008). An evaluation methodology for collaborative recommender systems. In *Proceedings of the International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution*, pages 224–231.

Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. (2011). Collaborative filtering recommender systems. In *Foundations and Trends in Human–Computer Interaction*, volume 4, pages 81–173.

[5]http://gtews.ime.usp.br - Accessed at December 22, 2015

Feldman, R. and Sanger, J. (2007). *he Text Mining Handbook: Advances Approaches in Analyzing Unstructured Data*. Cambridge University Press.

Flegel, U., Hoffmann, J., and Meier, M. (2010). Cooperation enablement for centralisticeary warning systems. In *InProceedings of the ACM SAC*, pages 2001–2008.

Font, F., Serra, J., and Serra, X. (2013). Folksonomy-based tag recommendation for collaborative tagging systems. In *International Journal on Semantic Web and Information Systems*, volume 9, pages 1–30.

Frei, S., May, M., Fiedler, U., and Plattner, B. (2006). Large-scale vulnerability analysis. In *SIGCOMM Workshop on LSAD*, volume 6, pages 131–138.

GCHQ and Cert-UK (2015a). Common cyber attacks: Reducing the impact. Technical report, The Information Security Arm of GCHQ & Cert-UK.

GCHQ and Cert-UK (2015b). Internet security threat report - 2014 trends. Technical report, Symantec.

Glauber, R., Loula, A., and Rocha-Junior, J. B. (2013). A mixed hybrid recommender system for given names. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 25–36.

Grobauer, B., Mehlau, J. I., and Sander, J. (2006). Carmentis: A co-operative approachtowards situation awareness and early warning for the internet. In *InIMF*, pages 55–66.

Herlocker, J., Konstan, J., Terveen, L., and Riedl, J. (2004). Evaluating collaborative filtering recommender systems. In *ACM Transactions on Information Systems (TOIS)*, pages 5–53.

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. In *10th Conference of Information and Knowledge Management*, pages 247–254.

Krsul, I. V. (1998). *Software Vulnerability Analysis*. PhD thesis, Purdue University, West Lafayette.

Ricci, F., Rokach, L., Shapira, B., and Kanto, P. B. (2011). *Recommender Systems Handbook*. Springer.

Santos, L. A. F., R., C., Gerosa, M. A., and Batista, D. M. (2013). Detecção de alertas de segurança em redes de computadores usando redes sociais. In *31.o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 791–803.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *10th international conference on World Wide Web*, pages 285–295.

Surjandari, I., Naffisah, M. S., and Prawiradinata, M. I. (2015). Text mining of twitter data for public sentiment analysis of staple foods price changes. In *Journal of Industrial and Intelligent Information*, volume 3, pages 253–257.

Zheng, N. and Li, Q. (2010). A recommender system based on tag and time information for social tagging systems. In *Expert Systems with Applications*, pages 4575–4587.

Abordagem autonômica para mitigar ciberataques em LANs

Luiz Arthur Feitosa dos Santos[1,2], **Rodrigo Campiolo**[1,2],*
Wagner Aparecido Monteverde[1], **Daniel Macêdo Batista**[2]

[1]Dept. Acad. Ciência da Computação – Universidade Tecnológica Federal do Paraná
[2]Instituto de Matemática e Estatística – Universidade de São Paulo

luizsantos@utfpr.edu.br,rcampiolo@utfpr.edu.br,
wagner.ap.monteverde@gmail.com e batista@ime.usp.br

Abstract. *It is a hard task for a human to deal with cyber attacks, mainly due the increasing number of users and heterogeneous devices in LAN encouraged by practices such as BYOD, that constantly brings new threats to these environments. Therefore, we propose an autonomic approach that uses machine learning to process, in a similar way of human memory, the historical of network usage and security alerts, to generate security rules, that are imposed to network using SDN resources, to mitigate cyber attacks. Experiments have shown that the approach was able to avoid 97.3% of malicious packets sent to a victim in DDoS TCP-SYN attacks and to mitigate 44 types of cyber threats, in a real network.*

Resumo. *Lidar com ciberataques em LANs tem se tornado uma árdua tarefa para o ser humano, principalmente devido a agregação crescente de usuários e dispositivos heterogêneos incentivada por práticas como BYOD, que trazem constantemente novas ameaças à segurança das LANs. Assim, propomos uma abordagem que emprega aprendizagem de máquina para processar, similarmente à memória humana, históricos de uso da rede e alertas de segurança, para extrair regras de segurança que são aplicadas autonomicamente na rede, por intermédio da tecnologia OpenFlow. Em ambiente simulado, a abordagem proposta foi capaz de mitigar 97,3% dos pacotes maliciosos em ataques DDoS TCP-SYN, já em uma rede real ajudou a mitigar 44 tipos de ciberameaças.*

1. Introdução

Práticas como *Bring Your Own Device* (BYOD) tornam as redes de computadores locais mais dinâmicas e amigáveis para o usuário final mas, em contrapartida, adicionam novas vulnerabilidades que são utilizadas frequentemente contra a segurança da rede [Symantec 2015]. O uso de mecanismos de segurança tradicionais, tais como *firewall* e *Intrusion Detection System* (IDS), normalmente exigem dedicação e perspicácia do administrador, que precisa constantemente analisar o histórico de uso da rede e alertas de segurança para manter um nível de segurança condizente com as ameaças. Todavia, muitas vezes o responsável pela rede local (*Local Area Network* – LAN) acumula cargos e por isto não tem tempo hábil para gerenciar a segurança de maneira apropriada. Em casos mais extremos, algumas LANs não possuem um responsável direto e/ou competente pela segurança do ambiente. Como consequência, a detecção e o tempo de reação às ameaças é

*Agradecimentos à Fundação Araucária, Secretaria de Estado da Ciência, SETI-PR, ao Governo do Estado do Paraná e a Rede Nacional de Pesquisa (RNP), pelo apoio a esta pesquisa e participação no evento. Agradecimento especial ao Romildo Martins da Silva (*In Memoriam*), pela contribuição.

muitas vezes elevado, o que pode comprometer o bom funcionamento da rede por longos períodos de tempo.

Há várias abordagens para facilitar o gerenciamento das redes e sua segurança. Por exemplo, tecnologias como Redes Definidas por Software (*Software-Defined Networks* – SDN) viabilizam o gerenciamento e acesso a novos fluxos de informação das redes. Com o OpenFlow [McKeown et al. 2008], um protocolo para SDN, é possível programar dinamicamente a rede para que *switches* insolem fluxos de redes, bloqueiem conexões indesejadas e implementem qualidade de serviço. Já a Computação Autonômica (CA) possibilita que sistemas se tornem auto-gerenciáveis, o que é possível através de ciclos autonômicos que monitoram, e caso necessário, adaptam o ambiente visando a proteção e a normalidade dos recursos sem a necessidade de interação humana [Hariri et al. 2006].

Neste contexto, este artigo apresenta uma abordagem autonômica que tem por objetivo retirar parte do ônus do administrador em manter a segurança de LANs. O presente trabalho inova pois propõe métodos que empregam algoritmos de análise de associação para processar históricos de uso da rede e alertas de segurança, extraindo então regras de segurança que são aplicadas autonomicamente na rede, por intermédio de SDN. Outra contribuição é a criação de um método inspirado na estrutura da memória humana, que permite gerar autonomicamente regras de segurança que: (a) mitigam problemas de segurança na rede; (b) previnem contra ameaças futuras; e (c) evitam que *hosts* e/ou serviços da rede sejam influenciados por ataques e pela própria defesa autonômica, em caso de falsos positivos.

Apresenta-se na Seção 2 os trabalhos relacionados, na Seção 3 a abordagem e métodos usados para implementá-la, na Seção 4 os experimentos usados para avaliar o desempenho da abordagem e os resultados obtidos, e na Seção 5 as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O conceito de SDN tem possibilitado novos tipos de aplicações para redes de computadores e segurança [Lara et al. 2013]. A reação autonômica contra ameaças aos recursos e serviços das redes também vem sendo abordada por diversos trabalhos [Yuan et al. 2014].

Dentre os trabalhos que utilizam OpenFlow para proteger redes, [Chung et al. 2013] propõem NICE, uma abordagem baseada em SDN para detecção e contenção de tentativas de comprometimento de máquinas virtuais que podem se tornar zumbis. NICE possui agentes baseados no IDS Snort que analisam o tráfego de cada servidor e realizam varreduras de vulnerabilidades; [Wang et al. 2013] propõem o NetFuse, um mecanismo para detectar e mitigar fluxos suspeitos que sobrecarregam redes que usam OpenFlow. Os fluxos identificados são roteados para um mecanismo de contenção, que pode, por exemplo, causar atrasos ou remoção de pacotes. Além disso, NetFuse possui um controle adaptativo inspirado em sistemas biológicos toxina-antitoxina, principalmente para lidar com falsos positivos; [Lara and Ramamurthy 2014] desenvolveram o OpenSec, que permite ao administrador especificar políticas de segurança reagindo a alertas de segurança usando OpenFlow. No OpenSec, o controlador encaminha os fluxos definidos nas políticas para unidades de processamento externas que realizam inspeção e devolvem alertas de segurança para o controlador selecionar uma reação.

Quanto aos trabalhos de CA que se destacam dentro do nosso contexto,

[Barrère et al. 2011, Barrère et al. 2012] afirmam que as redes autonômicas podem gerar novas vulnerabilidades e desta forma propõem um modelo para traduzir descrições de vulnerabilidades de elementos autonômicos em regras para políticas de segurança. Tais regras são interpretadas por um sistema autonômico que deve detectar vulnerabilidades e executar operações para sanar o problema; [Sibai and Menasce 2011, Sibai and Menasce 2012] propõem um arcabouço chamado AVPS, que tem por objetivo a auto-proteção contra ameaças internas que violem a política de segurança descrita pelo administrador da rede. O AVPS utiliza o Snort com políticas personalizadas para proteger sistemas contra abusos cometidos por usuários com privilégios ou contra configurações incorretas.

Já em relação a trabalhos que utilizam conjuntamente CA e SDN, [Chu et al. 2010] propuseram um método autonômico para mitigar ataques distribuídos de negação de serviço (*Distributed Denial of Service* – DDoS) por meio da contagem de volume de tráfego em *switches* OpenFlow; [Chen et al. 2014] introduziram um modelo de gerenciamento de segurança que emprega CA e OpenFlow para estimar, detectar e reagir prematuramente contra ataques à segurança em ambientes de Internet das Coisas (Internet *of Things* – IoT).

O presente trabalho se diferencia dos demais por propor o uso de CA e SDN para proteger LANs contra ciberataques. Este trabalho inova ao criar métodos que empregam aprendizagem de máquina não supervisionada para identificar padrões de uso e ataques à rede, para então gerar regras de segurança que mitigam problemas de segurança em LANs, sem intervenção humana. Não faz parte do escopo deste trabalho propor/melhorar mecanismos de detecção de intrusão. Trabalha-se com a hipótese de que os mecanismos de detecção atuais são eficientes e possuem uma taxa aceitável de falsos positivos.

2.1. Arquitetura Base

Em [Santos et al. 2014], propomos uma arquitetura autonômica para proteger LANs contra ciberataques (Figura 1). Tal arquitetura serve de base para o desenvolvimento deste trabalho e é implementada na ferramenta chamada *OpenFlow Intrusion Detection and Prevention System* (Of-IDPS) – arquitetura e o nome da ferramenta são usadas como sinônimos.

O processamento proposto na arquitetura, inicia com os sensores coletando dados a respeito do uso da rede e seus distúrbios (passo 1 da Figura 1). Estes dados vem do IDS Snort e dos *switches* OpenFlow. Na sequência o módulo Monitor normaliza os dados coletados pelos sensores (passo 2) e ajuda na identificação de alertas de segurança (passo 3), que podem ser gerados a partir da análise dos dados vindos do OpenFlow a respeito do uso da rede – os dados fornecidos pelo Snort já são considerados alertas de segurança e por isto só requerem normalização. Os alertas de segurança gerados são processados pelo módulo Análise e Planejamento, que cria planos de ações para remediar os problemas identificados (passo 4). O módulo Execução converte o plano de ações em regras de segurança, que são implantadas na rede com dois propósitos: mitigar fluxos de redes maliciosos, que já estão instalados e ativos na rede (passo 5.1); repudiar ou restringir a comutação de novos fluxos, quando estes estiverem relacionados com ataques à segurança (passo 5.2). O módulo Execução aplica as regras de segurança por intermédio dos atuadores (passo 6), que na prática são os *switches* e o controlador OpenFlow. Assim, os

fluxos legítimos são instalados normalmente na rede e fluxos maliciosos ou suspeitos são submetidos a contramedidas que visam resolver desequilíbrios na rede. Por fim, sensores retroalimentam a arquitetura (passo 1) fechando o ciclo autonômico infinito. Todas ações e informações geradas pela arquitetura são registradas no módulo Base de Conhecimento para auxiliar em tomadas de decisões futuras. Na Figura 1, a Base de Conhecimento está espalhada e representada pelos cilindros: Fluxos Normais, IDS, Regras Memória Sensorial, etc. Embora tenha-se usado Snort e o OpenFlow, é possível empregar outras tecnologias implementado novos sensores e atuadores.

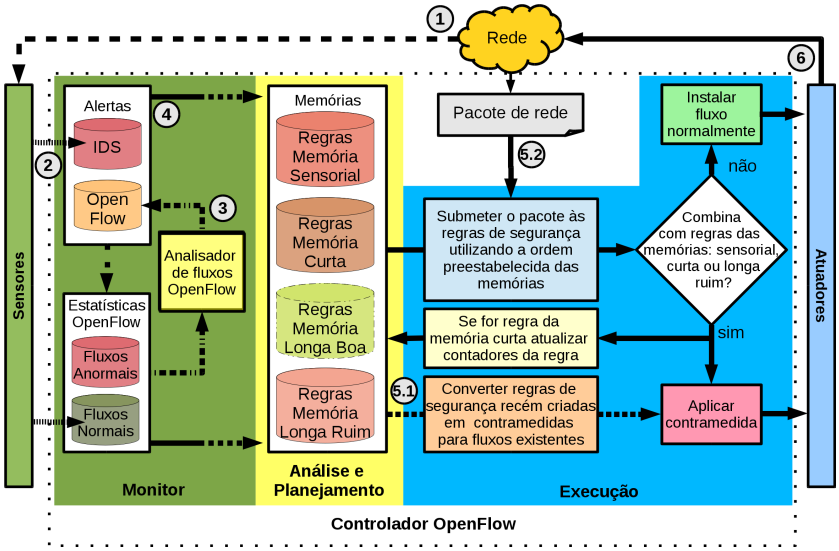


Figura 1. Arquitetura do Of-IDPS.

Em [Santos et al. 2014], o principal objetivo foi analisar e desenvolver ferramentas que permitissem monitorar e reagir contra ameaças que estivessem afetando a rede, ou seja, sensores, atuadores e os módulos Monitor e Execução. Já o objetivo do presente trabalho é explorar o módulo Análise e Planejamento adicionando o submódulo Memórias (Figura 1) que dá à arquitetura a capacidade de analisar o comportamento da LAN, aprendendo e distinguindo o tráfego de rede normal do malicioso, para autonomicamente privilegiar *hosts*/serviços legítimos e mitigar ciberameaças.

3. Metodologia

O principal objetivo deste trabalho é propor uma abordagem inspirada na estrutura da memória humana que possibilite autonomicamente criar e manter regras de segurança que levam em consideração o histórico de tráfego e alertas da rede.

3.1. Aprendizagem e Reação Autonômica

O módulo de Análise e Planejamento que faz parte da arquitetura apresentada na Figura 1, tem por objetivo analisar os dados do módulo Monitor, aprendendo com os padrões de uso da rede e ataques contra a LAN que fundamentarão a criação e manutenção de regras de segurança autonômicas que: (i) favoreçam fluxos de rede legítimos; e (ii) mitiguem fluxos de redes relacionados com ciberameaças.

Para que o Of-IDPS possa aprender com o histórico da rede, foi utilizado aprendizado de máquina não supervisionado por meio do algoritmo FP-Growth [Han et al. 2004], que permite descobrir/evidenciar relacionamentos entre informações em grandes volumes

de dados. Esta técnica é conhecida como análise de associação ou regras de associação, e uma das abordagens é minerar os conjuntos de itens frequentes (*frequent itemsets*) [Harrington 2012]. A escolha do FP-Growth é devido ao seu desempenho superior em relação a outros algoritmos similares [Han et al. 2004].

O FP-Growth tem como entrada um conjunto de itens referentes a transações de banco de dados e como saída os itens que mais aparecem juntos nestas transações. Além desses itens, também é devolvida outra informação chamada *suporte*, que é um valor numérico que representa a frequência/porcentagem de ocorrência de um conjunto de itens [Harrington 2012]. Assim, no contexto deste trabalho, é proposta a submissão do histórico da rede como transações a serem analisadas pelo FP-Growth, que então devolverá como saída: (i) padrões de ataques, fundamentados nos alertas de segurança; e (ii) padrões de uso da rede, embasado em dados que representam os fluxos de rede, obtidos dos *switches* OpenFlow.

Cada alerta de segurança a ser submetido no FP-Growth é composto pelos campos IP de origem, IP de destino, protocolo (TCP, UDP, ICMP, etc), porta de origem, porta de destino, identificação do alerta/problema de segurança e prioridade/risco do alerta. Os alertas, bem como o suporte mínimo esperado, são submetidos ao FP-Growth, que devolve conjuntos de itens que atendem ao suporte. Cada conjunto de itens obtido a partir dos alertas gerará uma regra de segurança. As regras de segurança possuem os mesmos campos dos alertas. Desta forma, para criar uma regra basta copiar os campos/itens do conjunto retornado para a regra de segurança. Todavia, em alguns casos, um conjunto pode não retornar todos os campos de um alerta, por não atender ao suporte. Caso isto ocorra, os campos faltantes são preenchidos com o valor especial *, que simboliza todo/qualquer, similar com o que acontece na especificação de regras de *firewalls*.

Por exemplo, se os alertas de segurança (fictícios) da Tabela 1 forem submetidos ao FP-Growth, com o suporte mínimo de 60%, o resultado será as regras de segurança da Tabela 2. A regra apresentada na linha 1 da Tabela 2, por exemplo, especifica que pacotes originados do endereço IP 20.0.0.1, destinados a qualquer endereço IP (*), utilizando o protocolo de rede TCP, originados de qualquer porta (*), destinados à porta 22, serão processados considerando o nível de prioridade de segurança médio.

Tabela 1. Exemplo de alertas a serem processados pelo FP-Growth.

N°	Endereço IP		Protocolo	Portas de rede		Alerta de segurança	
	Origem	Destino		Origem	Destino	Identificação	Prioridade
1	20.0.0.1	30.0.0.10	TCP	50000	22	SSH01	Média
2	20.0.0.1	30.0.0.20	TCP	50001	22	SSH02	Média
3	20.0.0.1	30.0.0.30	TCP	50002	22	SSH03	Média
4	20.0.0.1	30.0.0.40	TCP	50003	22	SSH01	Média
5	20.0.0.1	30.0.0.50	TCP	50004	22	SSH01	Média
6	20.0.0.1	30.0.0.60	TCP	50005	22	SSH01	Média
7	20.0.0.2	30.0.0.70	UDP	50006	53	DNS01	Alta
8	20.0.0.2	30.0.0.80	UDP	50007	53	DNS01	Alta
9	20.0.0.2	30.0.0.90	UDP	50008	53	DNS01	Alta
10	20.0.0.1	30.0.0.10	ICMP	0	1	ICMP01	Baixa

Tabela 2. Exemplo de regras criadas a partir dos alertas de segurança

N°	Endereço IP		Protocolo	Portas de rede		Alerta de segurança		Suporte (%)
	Origem	Destino		Origem	Destino	Identificação	Prioridade	
1	20.0.0.1	*	TCP	*	22	*	Média	60
2	20.0.0.1	*	TCP	*	*	*	Média	60
3	20.0.0.1	*	*	*	*	*	Média	70
4	*	*	TCP	*	22	*	Média	60
5	*	*	TCP	*	*	*	Média	60

Analisando as regras de segurança da Tabela 2, constata-se que existem alguns problemas com as regras de segurança geradas pelo FP-Growth. Assim, estes problemas e as respectivas soluções (representadas no Algoritmo 1), são apresentadas a seguir:

- Há regras conflitantes. Por exemplo, quando o controlador OpenFlow analisar um pacote (passo 5.2 da Figura 1) originado do *host* 20.0.0.1 destinado à porta TCP 22, haverá ambiguidade na escolha da regra a ser aplicada, pois todas combinam com o pacote. Para resolver este problema, aplica-se a regra que combina com a maior quantidade de campos do pacote e represente maior ameaça. Caso o impasse continue, aplica-se a primeira regra dentre as restantes;
- Algumas combinações de campos não são úteis. Por exemplo, não é possível formar regras somente com os campos: prioridade e/ou identificação. Também, regras contendo portas só devem ser utilizadas se estiverem acompanhadas do campo protocolo. O método *removerRegrasQuePossuemApenasCamposInvalidosParaFormacaoDeRegras()* do Algoritmo 1, remove regras compostas somente pelos campos identificação, prioridade, portas ou um combinado destes;
- O uso do FP-Growth pode gerar regras muito genéricas e afetar negativamente a rede inteira. Um exemplo é a regra 5 da Tabela 2, que se aplica a todos os fluxos TCP. Para resolver este problema propõem-se o uso de um suporte mínimo que varia em relação ao número de itens retornado pelo conjunto de itens (linhas 5-10 do Algoritmo 1). Então, exige-se que conjuntos com poucos itens tenham um suporte maior do que conjuntos com muitos itens. Um caso especial é quando a regra tem apenas o campo protocolo, pois como este tipo de regra é muito abrangente, então também são descartadas no método *removerRegrasQuePossuemApenasCamposInvalidosParaFormacaoDeRegras()* do Algoritmo 1. Desta forma, submetendo os alertas da Tabela 1 ao Algoritmo 1, a saída será apenas a regra 1 da Tabela 2, pois esta é a única que atende aos requisitos exigidos.

Portanto, o uso do Algoritmo 1 propicia a criação autonômica de regras de segurança no Of-IDPS. Entretanto, ainda há um grande problema, pois conforme o tempo passa e os alertas de segurança forem surgindo, a base de dados de alertas crescerá gradativamente, o que pode causar *overfitting* [Harrington 2012]. Ou seja, surge a possibilidade de que o grande número de alertas antigos ofusque os alertas novos, ao ponto de influenciar negativamente a criação de regras para mitigar distúrbios/ameaças recentes. A solução para este problema é apresentada na próxima subseção.

Algoritmo 1: gerarRegrasDeSegurançaAutonômicas()

```
   Entrada: listaAlertasSegurança - Lista de alertas gerados pelo Of-IDPS
   Saída: listaRegrasSegurança - Lista com regras de segurança
 1 listaConjuntoRegras ← φ
 2 listaRegrasSegurança ← φ
 3 listaConjuntoRegras ← processaAlertasComAlgoritmoFPGrowth(listaAlertasSegurança)
 4 listaConjuntoRegras ← removerRegrasQuePossuemApenasCamposInvalidosParaFormacaoDeRegras(listaConjuntoRegras)
 5 para cada regra em listaConjuntoRegras faça
 6     suporte ← regra.obterSuporte()
 7     numeroItens ← regra.obterNumeroItensPresenteNaRegra()
 8     percentualSuporte = 110 − 20 ∗ numeroItens
 9     se suporte ≥ percentualSuporte então
10         listaRegrasSegurança.adiciona(regra)
11 retorna listaRegrasSegurança
```

3.1.1. Reação Autonômica Inspirada na Memória Humana

Para que o Of-IDPS utilize informações de ataques do passado ou ocorrendo no presente, visando remediar problemas em andamento e prevenir novos ataques, foi desenvolvido um método inspirado na estrutura da memória humana. Isto porque, a memória humana é processada/dividida fundamentalmente em três partes: (1) sensorial, trata de informações ligadas aos sentidos do corpo (visual, tátil, etc) e que estejam ocorrendo no presente; (2) curta, armazena informações passadas ocorridas recentemente. (3) longa, armazena acontecimentos marcantes que ocorreram em um passado mais distante, se comparado com a memória curta [Baddeley 1997]. Então, a ideia é viabilizar ao Of-IDPS uma forma de processar o histórico da rede segundo a mesma política.

No Of-IDPS, a **memória sensorial** é responsável por processar os alertas de segurança extremamente recentes e que representam o que a rede está "sentindo" naquele exato momento. Dentre as memórias do Of-IDPS, esta é a única que não emprega o Algoritmo 1 para gerar as regras de segurança, pois deve atacar pontualmente o problema/alerta em curso. Assim, o processo de geração de regras na memória sensorial consiste em obter os alertas recém identificados pelo módulo Monitor, sendo que cada alerta obtido origina uma regra de segurança. A criação da regra consiste em copiar todos os campos do alerta para a regra de segurança, exceto pela porta do cliente que é generalizada (símbolo *). Isso é feito porque a porta do cliente é aleatória [Tanenbaum and Wetherall 2013], portanto é necessário generalizá-la com o intuito de mitigar também as novas investidas do *host* atacante. A análise e criação de regras na memória sensorial é executada continuamente, em um intervalo configurável de alguns poucos segundos. As regras de segurança geradas são armazenadas na base de dados Regras Memória Sensorial (Figura 1).

A **memória curta** do Of-IDPS retém alertas por pouco tempo. O processamento desta memória inicia com a obtenção de alertas ocorridos recentemente e registrados pelo módulo Monitor. Tais alertas são submetidos ao Algoritmo 1 para gerar regras de segurança extraídas dos padrões de ataques presentes nestes alertas. A cada ciclo autonômico das memórias sensorial e longa, as regras antigas são removidas e as novas incluídas. Todavia, o mesmo não pode ser feito na memória curta, pois as regras recém criadas mitigarão os fluxos maliciosos que geravam os alertas. Naturalmente, sem alertas o Of-IDPS presumirá que não há mais risco e destituirá as regras que policiavam o ataque, propiciando o ressurgimento do mesmo. Isto ocorre porque a memória curta analisa a situação de risco fundamentada em um histórico recente e breve, para tratar de problemas em andamento. Já a memória sensorial trata de situações imediatistas e pontuais e a memória longa é responsável por analisar todo o histórico de rede, que por sua vez não tende a mudar constantemente e nem rapidamente. Desta forma, a memória curta é a única que necessita de uma política de manutenção de regras de segurança mais elaborada. Assim, toda regra na base de dados Regras Memória Curta (Figura 1) tem um contador, que é incrementado caso a regra esteja em uso na rede ou decrementado se ociosa. A regra só é removida quando este contador atinge um valor mínimo, indicando que a regra não é mais útil. Este gerenciamento é feito na própria memória curta e com auxílio do módulo Execução, que fornece estatísticas a respeito do uso das regras (Figura 1).

A **memória longa** é responsável por aprender o que é "bom" ou "ruim" para a rede. Por isto, no Of-IDPS a memória longa é dividida em duas: (1) **longa ruim**: tem

por objetivo identificar as principais ameaças que afligem a rede. Para isto, o histórico de alertas é submetido ao Algoritmo 1. As regras de segurança geradas são gravadas na base de dados Regras Memória Longa Ruim; (2) **longa boa**: ajuda a identificar quais são os principais serviços e *hosts* da rede. Para tanto, o módulo Monitor distingue os fluxos de rede OpenFlow que estão relacionados com alertas de segurança dos que não estão, isto é representado respectivamente pelas bases Fluxos Anormais e Fluxos Normais, na Figura 1. Os campos extraídos da base Fluxos Normais seguem o mesmo padrão dos alertas de segurança, com exceção dos campos identificação e prioridade que sinalizam que estes são fluxos legítimos, e não ameaças. Assim, o processamento da memória longa boa consiste em analisar a base Fluxos Normais, utilizando o Algoritmo 1 e gravar as regras de segurança geradas na base Regras Memória Longa Boa. Outra diferença é que os pacotes que combinarem com estas regras são instalados na rede sem nenhuma restrição.

As regras de segurança geradas pelas memórias são armazenadas em suas respectivas bases de dados e aplicadas na rede pelo módulo Execução (passos 5.1 e 5.2 da Figura 1). Durante a comutação, no controlador OpenFlow, cada pacote é comparado com as regras de segurança das memórias do Of-IDPS. Neste processo, a primeira memória que tiver uma regra que combine com o pacote será aplicada e a respectiva contramedida é adotada. Dependendo da contramedida [Santos et al. 2014], há três possibilidades de encaminhamento: (1) sem restrições (regras da memória longa boa); (2) com restrições de taxa de transferência de dados (para regras com prioridade de segurança baixa e média); (3) ou bloquear completamente (regras com prioridade alta). Caso o pacote não combine com as regras de nenhuma memória, este é encaminhado normalmente.

Os pacotes comutados obrigatoriamente são submetidos à seguinte ordem de análise entre as memórias: sensorial, curta e longa ruim. Contudo, a ordem de processamento da memória longa boa em relação às outras memórias é configurável. Isto foi feito pensando-se em deixar o Of-IDPS flexível para adequar-se a diferentes tipos/necessidades de redes e administradores. Assim, são possíveis três configurações quanto à ordem da memória boa e cada uma tem prós e contras: (1) com a memória longa boa sendo processada antes da sensorial, os fluxos bons/comuns não são afetados pelas contramedidas das outras memórias, evitando problemas de falsos positivos. Todavia, se ocorrerem ataques nos fluxos "bons", estes não serão imediatamente mitigados, pois estarão amparados pela memória longa boa; (2) caso a memória longa boa seja posta antes da memória curta, os ataques muito recentes serão combatidos pela memória sensorial, mesmo que ocorram em fluxos considerados "bons", mas só de forma pontual, pois não serão aplicadas as regras da memória curta. (3) usando a memória longa boa antes da longa ruim, o Of-IDPS inicialmente terá conhecimento prévio do que é bom ou ruim (memórias longa boa e ruim). Porém, o Of-IDPS não hesitará em reagir (memórias sensorial e curta) contra fluxos ordinários (memória longa boa), caso estes representem riscos a rede. Se o problema persistir durante muito tempo, o Of-IDPS aprenderá e mudará sua conclusão a respeito do que é considerado bom ou ruim, tal como na memória humana. É importante ressaltar que a troca da ordem de análise da memória longa boa, só afeta a forma de reação frente aos fluxos considerados ordinários, não afetando a capacidade do Of-IDPS em combater fluxos de redes considerados incomuns.

4. Experimentos e Resultados

Os experimentos apresentados nesta seção visam avaliar a reação autonômica do Of-IDPS, utilizando os métodos apresentados na Seção 3, frente a incidentes de segurança. A avaliação da reação autonômica proposta foi realizada em dois ambientes: (1) LAN simulada; (2) LAN real. A escolha de dois ambientes é devido ao fato que o ambiente virtual proporciona experimentos mais precisos e sem interferências externas, já o ambiente real permite analisar o comportamento do Of-IDPS em uma rede em produção.

O Of-IDPS foi desenvolvido utilizando o arcabouço de controlador OpenFlow Beacon, escolhido por apresentar melhor desempenho se comparado com outros controladores OpenFlow conhecidos [Erickson 2013]. Diferentemente de trabalhos similares, o código-fonte do Of-IDPS está disponível publicamente em `https://github.com/luizsantos/Of-IDPS` sob licença GNU GPLv3[1]. Para estes experimentos, o Of-IDPS foi configurado para utilizar a seguinte ordem de análise de regras entre as memórias: 1ª – sensorial, 2ª – curta, 3ª – longa boa e 4ª – longa ruim. O tempo de retenção dos últimos alertas identificados para as memórias sensorial e curta, são respectivamente 3 e 30 segundos[2].

Na LAN simulada foram realizados experimentos com ataques DDoS TCP-SYN. A simulação é feita utilizando Mininet [Lantz et al. 2010, Antonenko and Smelyanskiy 2013]. Durante o ataque, a vítima executa um servidor HTTP Apache[3] e o ataque DDoS com origem desconhecida (*spoof*) é feito com a ferramenta Hyenae[4]. Foram realizados 3 experimentos variando a quantidade de pacotes maliciosos enviados para a vítima, sendo estas as quantidades: 5.000, 10.000 e 20.000. O objetivo destes experimentos é observar se o Of-IDPS apresenta um comportamento uniforme, na mitigação de ataques, independente do número de pacotes maliciosos tratados. Cada experimento é composto por 15 repetições e foi executado em cenários de rede com e sem o Of-IDPS.

A Tabela 3 apresenta os dados gerados por cada um dos experimentos com DDoS. Analisando principalmente a última coluna da tabela, que apresenta a relação entre o número de pacotes tratados pelas vítimas com e sem o Of-IDPS, é possível constatar que a abordagem proposta foi capaz de mitigar os ataques em todos os experimentos. Por exemplo, no experimento com 10.000 pacotes na rede sem o Of-IDPS (experimento de controle) o atacante enviou 10.019 pacotes, os quais geraram 19.908 pacotes de resposta da vítima. A vítima gerou quase o dobro de pacotes em relação ao atacante, pois o atacante utilizou endereços IPs falsos e aleatórios (*spoof*) para efetuar o ataque DDoS, ou seja, os pacotes de resposta não retornam para o atacante. No experimento com 10.000 pacotes, na rede com Of-IDPS, o atacante enviou 10.200 pacotes e a vítima apenas 1.008 pacotes. Portanto, nos experimentos com 10.000 pacotes, a vítima da rede com Of-IDPS teve uma diminuição de 94,9% dos pacotes maliciosos tratados, se comparada à vítima da rede sem Of-IDPS.

Constatou-se também que a variação do número de pacotes maliciosos enviados durante os ataques DDoS não influenciam drasticamente o comportamento da arquitetura.

[1] `http://www.gnu.org/licenses/` - acessado em 03 de dezembro de 2015.

[2] O uso destes tempos é fundamentado em extensivos testes, que infelizmente não cabem neste artigo.

[3] `http://www.apache.org/` - acessado em 03 de dezembro de 2015.

[4] `http://sourceforge.net/projects/hyenae/` - acessado em 03 de dezembro de 2015.

Por exemplo, os gráficos da Figura 2 apresentam o fluxo médio (linha) e o desvio padrão (sombra) dos pacotes enviados/recebidos pelo atacante e vítima. Analisando os dados do atacante (linha laranja com quadrado) e vítima (linha vermelha com círculo) da rede sem Of-IDPS, verifica-se a contundência dos ataques DDoS, pois a rede é inundada pelos pacotes maliciosos e, por consequência, os recursos da rede e da vítima são comprometidos. Já examinando os dados do acatante (linha azul com asterisco) e vítima (linha verde com triângulo) que utilizam o Of-IDPS, é possível comprovar que a criação de regras autonômicas proposta consegue mitigar os ataques, pois com cerca de 5 segundos de seu início as medidas reativas começam a surtir efeito, restringindo a taxa de transferência de dados dos pacotes maliciosos, o que notoriamente ajuda a conservar recursos da rede e proteger a vítima. Então, conclui-se que o número de pacotes maliciosos enviados durante os ataques não geram distúrbios que invalidam o comportamento da arquitetura, nem a mitigação do problema.

Tabela 3. Dados dos experimentos com ataques DDoS.

Número de pacotes maliciosos enviados	Número pacotes LAN sem Of-IDPS		Número pacotes LAN com Of-IDPS		Comparação entre vítimas (%)*
	Atacante	Vítima	Atacante	Vítima	
5.000	5.020	9.965	5.017	957	90,4
10.000	10.019	19.908	10.200	1.008	94,9
20.000	20.069	39.886	20.323	1.080	97,3

* - Porcentagem de pacotes que a vítima com Of-IDPS deixou de tratar em relação à vítima da rede sem Of-IDPS.
Observação - Durante os experimentos são utilizados pacotes ICMP *echo request/reply* para identificar o inicio e o final de cada teste e também são produzidos naturalmente pacotes ARP durante a comunicação na LAN. Devido a isto há um pequeno acréscimo na quantidade de pacotes em cada experimento.

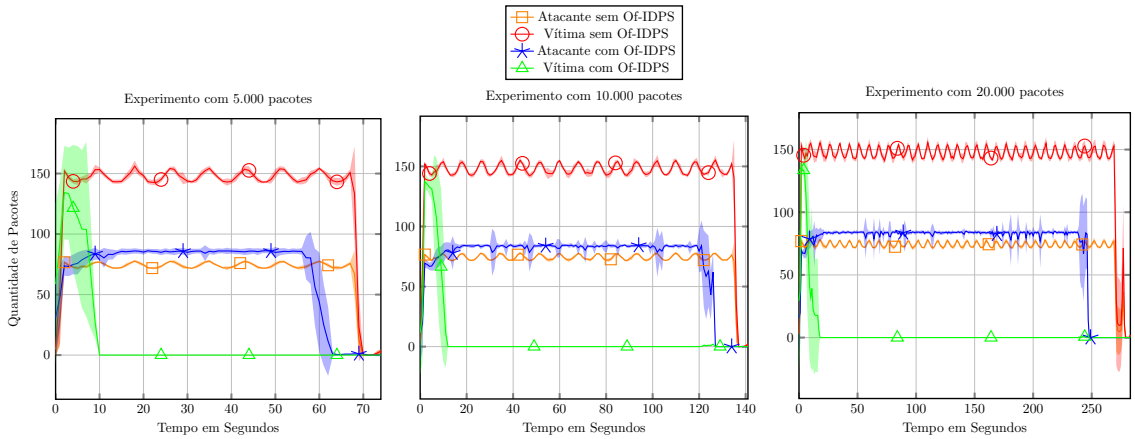


Figura 2. Experimentos variando a quantidade de pacotes no ataque DDoS.

Complementarmente, foram efetuados experimentos com o Of-IDPS em uma LAN real, para verificar a reação autonômica proposta frente às adversidades de um ambiente em produção. Mais especificamente os objetivos foram: (1) verificar a reação do Of-IDPS frente a outros problemas de segurança, além dos já avaliados em [Santos et al. 2014] e do ataque DDoS apresentado anteriormente; (2) analisar se as respostas autonômicas contra problemas de segurança não afetam negativamente *hosts*/serviços de redes legítimos; (3) avaliar a eficácia das memórias longa boa e ruim em utilizar o histórico da rede para identificar os principais *hosts*/serviços e ataques da LAN.

Desta forma, o Of-IDPS foi instalado na LAN do Grupo de Trabalho Early Warning System (GT-EWS) da Rede Nacional de Ensino e Pesquisa (RNP). Tal LAN fornece infraestrutura para o funcionamento de um sistema de alertas antecipados fundamentados principalmente em mensagens de segurança postadas em redes sociais [Santos et al. 2013], e que pode ser acessado via Internet em `http://gtews.cm`.

`utfpr.edu.br/ews/`. Esta LAN é basicamente composta por: (a) um Controlador OpenFlow no qual está instalado o Of-IDPS; (b) um *host* responsável por coletar postagens no Twitter e Facebook; (c) um servidor Xen[5], com máquinas virtuais que dão suporte aos serviços do projeto, tal como: banco de dados, *Web service*, servidor HTTP, IDS, núcleo de processamento do sistema de alertas antecipados, etc.

A LAN do GT-EWS está conectada na LAN da Universidade Tecnológica Federal do Paraná (UTFPR), que por sua vez está ligada à Internet. Então, é função do Of-IDPS proteger a LAN contra ciberameaças provindas da própria LAN do GT-EWS, da LAN da UTFPR e da Internet.

Para a visualização do comportamento do Of-IDPS na rede, a equipe do GT-EWS desenvolveu uma interface Web que permite ver em tempo de execução as regras de segurança presentes nas memórias do Of-IDPS. Dois fragmentos desta interface são apresentados na Figura 3. Basicamente a interface Web possui um menu lateral esquerdo que dá acesso a cada uma das memórias (ver Figura 3) e uma tabela para visualização dos alertas. Cada linha da tabela é um alerta e apresenta: Prioridade de segurança (*Priority*), identificação do problema que originou a regra (*Descrip.*), IP de origem (*Source*), IP de destino (*Dest.*), protocolo (*Proto.*), porta de origem (*T.Source*), porta de destino (*T.Dest*), suporte (*Support*) e contador de vida (*Life* – só utilizado pela memória curta).

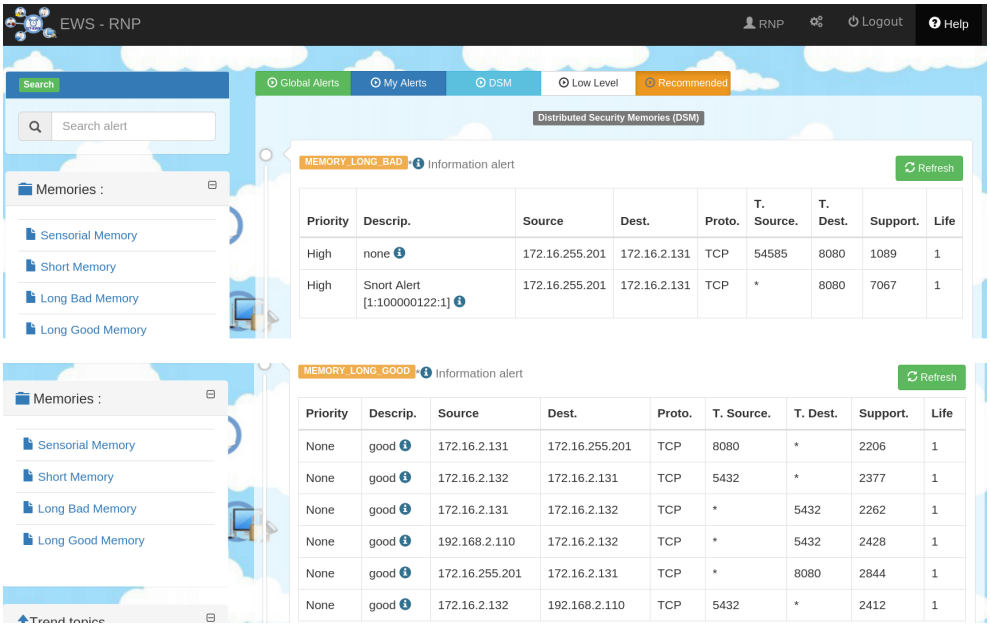


Figura 3. Memórias longa (a) ruim e (b) boa na interface Web do GT-EWS.

O Of-IDPS está em execução na LAN do GT-EWS há 6 meses. Neste período o módulo Monitor (Figura 1) identificou 44 tipos de ciberameaças, que deram origem a 15.098 alertas de segurança. Todo este histórico de alertas processado pela metodologia da memória longa ruim resulta nas duas regras autonômicas presentes na tabela da Figura 3-a. A primeira regra bloqueia pacotes originados de 172.16.255.201, destinados a 172.16.2.131, usando o protocolo TCP, originados da porta cliente 54585 e destinados a porta 8080. Esta regra foi formada por alertas que representam mais de um

[5]`http://xenserver.org/` - acessado em 03 de dezembro de 2015.

tipo de problema de segurança, por isto a generalização (*none*) do campo identificação (*Descrip.*). No Of-IDPS pacotes com alta (*high*) prioridade de segurança são bloqueados [Santos et al. 2014]. A segunda regra faz basicamente o mesmo que a primeira, mas generalizando a porta de origem (*) e especificando o tipo de ataque (*Snort Alert [1:100000122:1]*). Ambas regras condizem com o cenário de rede analisado, pois devido à estrutura da LAN da UTFPR, todo acesso provindo da Internet chega na LAN do GT-EWS por intermédio do *host* 172.16.255.201. Também, a maior parte dos acessos da Internet são destinados ao servidor HTTP (*host* 172.16.2.131). Então, na LAN do GT-EWS, a memória longa ruim aprendeu que muitos ataques ocorrem partindo do *host* 172.16.255.201, para 172.16.2.131 e estas ciberameaças atentam principalmente contra vulnerabilidades do servidor HTTP Apache (*Snort Alert [1:100000122:1]*), na porta TCP/8080 e por isso devem ser bloqueadas.

Neste mesmo período de 6 meses, foram também instalados nos *switches* OpenFlow, 10.205.790 fluxos de rede considerados idôneos (base Fluxos Normais da Figura 1). A análise destes fluxos pela memória longa boa pode ser vista na Figura 3-b. A primeira e a penúltima regra indicam que é comum a criação de fluxos de rede entre os *hosts* 172.16.2.131 e 172.16.255.201, usando a porta TCP/8080. Na interface, o campo *Priority* na memória boa é marcado como *None*, simbolizando que não há prioridade/risco de segurança para os pacotes que combinam com estas regras. Já o campo identificação (*Descrip.*) é preenchido com *good*, representando que a regra foi gerada pela memória longa boa. As demais regras demonstram que é ordinário o acesso dos *hosts* 172.16.2.131 (Web *Service* e servidor HTTP) e 192.168.2.110 (Controlador OpenFlow) ao *host* 172.16.2.132 na porta TCP/5432 (banco de dados). As regras criadas pela memória longa boa, também são condizentes com o uso da LAN do GT-EWS, pois nesta rede é muito comum que o *host* 172.16.255.201 acesse o servidor HTTP, que por sua vez, acessa o *host* do banco de dados. Já o controlador OpenFlow, que representa o núcleo do Of-IDPS, armazena as regras de segurança geradas por este, no *host* do banco de dados, para que sejam visualizadas posteriormente no servidor HTTP.

Para possibilitar uma investigação mais minuciosa a respeito da segurança da rede do GT-EWS durante os testes com o Of-IDPS, todos os *hosts* foram monitorados adicionalmente pelas ferramentas OSSEC-HIDS[6] e CACTI[7]. Todos os dados obtidos a respeito da segurança da LAN passaram ainda pela investigação constante dos autores deste trabalho. Todavia, em nenhum momento foi constatado o comprometimento da segurança da LAN do GT-EWS, destacando que a abordagem proposta consegue efetivamente mitigar ciberameaças sem afetar negativamente os serviços legítimos da LAN.

Além dos experimentos com DDoS apresentados neste trabalho, também foram executados testes em menor escala, que empregaram 50.000 e 100.000 pacotes maliciosos. Tais testes tiveram resultados bem similares aos apresentados no artigo. Inclusive o Of-IDPS tende a apresentar resultados ligeiramente melhores ao se aumentar o número de pacotes maliciosos no ataque DDoS, pois os fluxos do ataque ficam com sua taxa de transferência de dados reduzida por mais tempo. Isto pode ser observado na Tabela 3 e Figura 2. Em [Santos et al. 2014], a arquitetura mitigou 89,05% dos pacotes DoS com origem conhecida (IP do atacante). Já no presente trabalho, conseguiu-se mitigar em média

[6] `http://ossec.github.io/` - acessado em 03 de dezembro de 2015.

[7] `http://www.cacti.net/` - acessado em 03 de dezembro de 2015.

94,2% dos pacotes maliciosos em ataques DDoS com origem desconhecida. Ou seja, a nova abordagem conseguiu melhorar 5,15 pontos percentuais em um cenário de ataque mais complexo. Ainda nos experimentos com DDoS, observa-se que o início da reação autonômica contra o ataque foi de 5 segundos e a mitigação do problema se dá com 10 segundos. Tal resultado é considerado aceitável, pois em [Chen et al. 2014] a mitigação do mesmo problema se dá com 40 segundos. Todavia, ambos resultados são satisfatórios se comparados ao tempo que um administrador humano levaria para identificar e mitigar o problema. Ataques que ocorram em menos de 5 segundos podem burlar as memórias sensorial e curta, contudo serão mitigados pela memória longa ruim, caso persistam.

5. Conclusões e Trabalhos Futuros

Os experimentos nas LANs simuladas e real demonstram a efetividade da abordagem proposta em mitigar problemas de segurança. Por exemplo, o Of-IDPS manteve a integridade da LAN do GT-EWS, mesmo depois da investida de 44 tipos de ciberameaças. Nos experimentos com DDoS, a reação autonômica proposta conseguiu mitigar até 97,3% dos pacotes maliciosos, na vítima. Também, constatou-se que a reação autonômica proposta não influencia negativamente os serviços legítimos da rede, principalmente devido ao uso da memória longa boa. Isto pode ser observado na Figura 3-a, que apresenta as regras autonomicamente criadas para bloquear fluxos destinados ao servidor HTTP, que notoriamente traz riscos à rede. Contudo, dificilmente um administrador de rede bloquearia um serviço que é vital para o projeto do GT-EWS, como é o caso. Então, similarmente ao administrador humano, a memória longa boa identifica que o servidor HTTP é um serviço ordinário, e permite seu uso na rede (regras 1 e 5 da Figura 3-b), se contrapondo à memória longa ruim, o que evidencia que as memórias longa boa e ruim conseguem avaliar o histórico da rede e identificar os principais *hosts*/serviços e ataques na LAN.

Assim, por meio da abordagem proposta e dos testes realizados demonstrou-se que é possível criar métodos para auto-proteger LANs de ciberataques, com o mínimo possível de intervenção humana. Como trabalhos futuros pretende-se correlacionar o histórico da LAN com informações externas a respeito de ciberameaças.

Referências

Antonenko, V. and Smelyanskiy, R. (2013). Global network modelling based on mininet approach. In *Proceedings of the 2nd ACM SIGCOMM*, HotSDN '13, pages 145–146, NY, USA. ACM.

Baddeley, A. (1997). *Human Memory: Theory and Practice*. Psychology Press.

Barrère, M., Badonnel, R., and Festor, O. (2011). Supporting vulnerability awareness in autonomic networks and systems with oval. In *Proceedings of the 7th International CNSM'11*, pages 357–363, Laxenburg, Austria. International Federation for Information Processing.

Barrère, M., Badonnel, R., and Festor, O. (2012). Collaborative remediation of configuration vulnerabilities in autonomic networks and systems. In *CNSM'12, 8th international conference and 2012 workshop on SVM*.

Chen, Q., Abdelwahed, S., and Erradi, A. (2014). A model-based validated autonomic approach to self-protect computing systems. *IEEE Internet of Things Journal*.

Chu, Y., Tseng, M., Chen, Y., Chou, Y., and Chen, Y. (2010). A novel design for future on-demand service and security. In *12th ICCT IEEE*, pages 385–388.

Chung, C.-J., Khatkar, P., Xing, T., Lee, J., and Huang, D. (2013). Nice: Network intrusion detection and countermeasure selection in virtual network systems. *Dependable and Secure Computing, IEEE Transactions on*, 10(4):198–211.

Erickson, D. (2013). The beacon openflow controller. In *Proceedings of 2th ACM SIGCOMM*, HotSDN '13, pages 13–18, NY, USA. ACM.

Han, J., Pei, J., Yin, Y., and Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87.

Hariri, S., Khargharia, B., Chen, H., Yang, J., Zhang, Y., Parashar, M., and Liu, H. (2006). The autonomic computing paradigm. *Cluster Computing*, 9(1):5–17.

Harrington, P. (2012). *Machine Learning in Action*. Manning Publications Co.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM*, Hotnets-IX, pages 1–6, NY, USA. ACM.

Lara, A., Kolasani, A., and Ramamurthy, B. (2013). Network innovation using openflow: A survey. *Communications Surveys Tutorials, IEEE*, PP(99):1–20.

Lara, A. and Ramamurthy, B. (2014). Opensec: A framework for implementing security policies using openflow. In *Proceddings of Globecom 2014 - Communication and Information System Security Symposium*, pages 781–786.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Santos, L. A. F., Campiolo, R., and Batista, D. M. (2014). Uma arquitetura autonômica para detecção e reação a ameaças de segurança em redes de computadores. In *Anais do 4o Workshop em Sistemas Distribuídos Autonômicos, SBRC 2014*, pages 45–48.

Santos, L. A. F., Campiolo, R., Gerosa, M. A., and Batista, D. M. (2013). Deteção de alertas de segurança em redes de computadores usando redes sociais. In *Anais do XXXI SBRC*, pages 791–804.

Sibai, F. M. and Menasce, D. (2011). Defeating the insider threat via autonomic network capabilities. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–10.

Sibai, F. M. and Menasce, D. (2012). Countering network-centric insider threats through self-protective autonomic rule generation. In *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*, pages 273–282.

Symantec (2015). Internet security threat report 2015. Technical Report 20, Symantec.

Tanenbaum, A. S. and Wetherall, D. J. (2013). *Computer Networks*. Pearson Education.

Wang, Y., Zhang, Y., Singh, V., Lumezanu, C., and Jiang, G. (2013). Netfuse: Short-circuiting traffic surges in the cloud. In *2013 IEEE ICC*, pages 3514–3518.

Yuan, E., Esfahani, N., and Malek, S. (2014). A systematic survey of self-protecting software systems. *ACM Trans. Auton. Adapt. Syst.*, 8(4):17:1–17:41.

Trilha Principal do SBRC 2016
Sessão Técnica 15
Redes de Sensores

Agregação de Dados com Desvio de Buracos para Redes de Sensores Sem Fio com *Sinks* de Alto Alcance

Moysés M. Lima[1], Horácio A. B. F. de Oliveira[1], Eduardo F. Nakamura[1] Raimundo Barreto[1] e Daniel L. Guidoni[2]

[1]Instituto de Computação – Universidade Federal do Amazonas
[2]Departamento de Computação – Pontifícia Universidade Católica de Minas Gerais
{moyses.lima,horacio,nakamura,rbarreto}@icomp.ufam.edu.br, guidoni@ufsj.edu.br

Abstract. *In Wireless Sensor Networks (WSNs), Greedy Forwarding is a common technique used by most geographic routing algorithms. It uses location information of neighbors to choose the next hop geographically closer to the destination node. However, this strategy requires previous execution of a location system which is not always available in forest scenarios and in most cases is not able to overcome routing holes. In this work, we develop a geographic routing algorithm that performs data aggregation and is able to overcome routing holes in WSNs. Our approach, called ARESTA, uses a sink with long-range communication that sends an advertisement package to the entire network in a single hop. The ARESTA algorithm is also able to overcome holes through a new technique that uses RSSI (Received Signal Strength Indicator) to detect and bypass a hole. The RSSI values sent by the long-range sink node are used to create a time metric to guide in-network data aggregation. Our simulations show that the proposal has high delivery rate, scalability on densely deployed wireless networks, and also can identify and detect holes with all of the benefits of GF algorithms.*

Resumo. *Em Redes de Sensores sem Fio (RSSFs), a técnica de encaminhamento guloso é bastante utilizada pela maioria dos algoritmos de roteamento geográfico. A abordagem utiliza a informação de localização dos vizinhos para escolher o próximo salto, geograficamente mais próximo do destino. No entanto, a estratégia requer a utilização prévia de um sistema de localização, que nem sempre está disponível em cenários de floresta e na maioria dos casos não é capaz de lidar com buracos. Neste trabalho, desenvolvemos um algoritmo de roteamento geográfico que realiza agregação de dados e é capaz de desviar buracos de roteamento em RSSFs. Nossa abordagem, chamada ARESTA, utiliza um sink com alto alcance de comunicação que envia um pacote de anúncio a toda rede em um único salto. O algoritmo ARESTA é capaz de lidar com buracos de roteamento por meio de uma nova técnica que utiliza o RSSI (Received Signal Strength Indicator) dos nós para detectar e desviar buracos. Os valores do RSSI divulgados pelo nó sink de longo alcance são utilizados para criar um temporizador que auxilia na agregação de dados. As simulações realizadas mostram uma eficiente taxa de entrega, escalabilidade em redes densamente populadas, além de identificar e detectar buracos mantendo os benefícios de algoritmos de encaminhamento guloso.*

1. Introdução

Uma Rede de Sensores sem Fio (RSSF) consiste em um conjunto de nós sensores depositados em uma área de interesse que de forma colaborativa monitoram condições ambientais físicas e químicas em diferentes locais [Akyildiz et al. 2002b]. Por esta razão, as RSSFs são essenciais no monitoramento de condições ambientais tais como temperatura, luminosidade, movimento e presença de certos tipos de objetos. Na maior parte das aplicações, os nós sensores possuem limitações energéticas e de largura de banda, fazendo com que técnicas de otimização do consumo de energia e comunicação sejam necessárias.

Quando consideramos o número de saltos entre nós regulares e o nó *sink* em uma RSSF, podemos classificar as abordagens como redes de único salto e redes de múltiplos saltos [Akyildiz et al. 2002a]. Na primeira classificação, os nós regulares enviam suas informações diretamente ao *sink*, simplificando a implementação de soluções de envio de pacotes. Entretanto, é necessária a utilização de um dispositivo de comunicação sem fio de longo alcance, que normalmente é custoso em termos de consumo energético e hardware apropriado. Já na abordagem em múltiplos saltos, os nós regulares enviam suas informações ao *sink* utilizando comunicação de curto alcance estabelecida com um ou mais vizinhos em localização intermediária.

Avanços na área de comunicação sem fio demonstram que a propagação do sinal de longo alcance se mostra confiável e pode ser utilizada para estimar distâncias em uma RSSF. Tal estimativa é baseada nos valores de RSSI (Received Signal Strength Indicator) dos nós envolvidos na comunicação, considerando a relação entre a força do sinal recebido e a distância como uma função de distância entre o emissor e o receptor [Al Alawi 2011]. Como apresentado em [Lima et al. 2013], um possível cenário para a implementação destas tecnologias é o projeto ATTO (Amazonian Tall Tower Observatory), que consiste em uma torre de 320 m de altura e alta capacidade de comunicação, cujo objetivo é mapear eventos e fornecer medidas confiáveis de gases do efeito estufa [Tollefson 2010]. O sinal emitido pelo dispositivo de comunicação instalado na torre poderia alcançar um grande número de nós, como apresentado na Figura 1.

Os protocolos de roteamento geográficos têm sido vistos como uma das principais soluções de roteamento em Redes de Sensores Sem Fio (RSSFs) por serem escaláveis, dinâmicos e possuírem uma alta taxa de entrega de dados [Kim et al. 2005, You et al. 2009]. Contudo, em algumas situações os métodos envolvendo informações geográficas não podem ser utilizados, especialmente em cenários onde a informação de

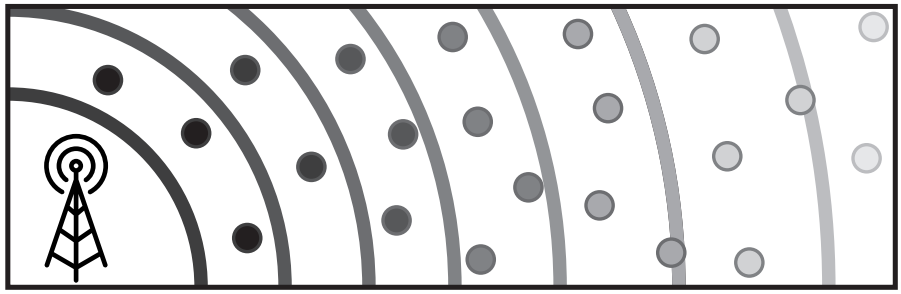


Figura 1. Valores de RSSI decrescendo à medida que a consulta do *sink* de alto alcance se propaga na rede. Nós mais distantes do *sink* receberão a mensagem com potência menor.

localização não está disponível ou não existe [Éfren Souza et al. 2012].

O encaminhamento guloso é a forma mais simples de se implementar o roteamento geográfico e é particularmente atraente nas redes de sensores. A técnica utiliza uma estratégia simples de geração de rotas em que um nó sensor sempre encaminha um pacote ao vizinho geograficamente mais próximo ao nó *sink*. Um desafio importante na utilização dessa técnica é a ocorrência de buracos de roteamento durante o envio dos pacotes ao *sink*. Os buracos são normalmente associados à regiões onde os nós estão impossibilitados de encaminhar adiante informações recebidas. Sua ocorrência pode ser causada por obstáculos físicos, esgotamento energético, falhas na comunicação entre vizinhos eleitos, implantação incorreta e devido ao fenômeno dos mínimos locais [Chen and Varshney 2007].

Um outro desafio na utilização de algoritmos de roteamento geográfico é a necessidade de localização dos nós, que pode ser custosa e susceptível a grandes erros em RSSFs [Éfren Souza et al. 2012]. Além disso, a capacidade de processamento dos nós pode ser utilizada como estratégia para a redução do consumo de energia, durante o roteamento. Associada à solução de encaminhamento, a agregação de dados pode melhorar significativamente o desempenho de um algoritmo de encaminhamento guloso, pois provê a combinação de vários pacotes de dados de entrada em um único pacote de informação relevante.

Neste trabalho, propomos um algoritmo de roteamento geográfico, que provê agregação de dados e é capaz de lidar com buracos de roteamento em RSSFs, sem a necessidade de utilização de informações de localização, chamado ARESTA. A proposta é uma melhoria do algoritmo apresentado em [Lima et al. 2013], chamado RADR (Roteamento e Agregação de Dados baseado no RSSI). A idéia principal do algoritmo é aproveitar o grande alcance do nó *sink*, equipado com um dispositivo de comunicação de maior potência, de tal forma que em um único salto todos os nós da rede sejam alcançados. O retorno dos nós sensores para o nó *sink* é realizado através de múltiplos saltos, utilizando uma nova técnica de encaminhamento guloso que utiliza os valores do RSSI (*Received Signal Strength Indicator*) para calcular o próximo salto em direção ao nó *sink*, calcular o tempo de espera que possibilita a agregação de dados e detectar buracos de roteamento durante o processo de encaminhamento.

O restante deste trabalho está organizado da seguinte forma: na Seção 2, apresentamos e classificamos os trabalhos que serviram como base de pesquisa para o presente artigo, enquanto na Seção 3 descrevemos o algoritmo ARESTA, cuja avaliação de performance é mostrada na Seção 4. Uma breve discussão sobre a aplicabilidade da proposta é apresentada na Seção 5 e, finalmente, na Seção 6 apresentamos nossas conclusões e trabalhos futuros.

2. Trabalhos Relacionados

A técnica de encaminhamento guloso tem sido bastante utilizada pela maioria dos algoritmos de roteamento geográfico por se tratar de uma abordagem simples, eficiente e escalável [Boukerche et al. 2008]. A abordagem utiliza a informação de localização dos vizinhos para encaminhar pacotes ao nó que está geograficamente mais próximo do destino. Visando facilitar o entendimento sobre nossa proposta, classificamos os algoritmos de roteamento geográfico em três categorias principais de algoritmos: (1) baseados em

coordenadas geográficas; (2) baseados em coordenadas virtualis; e (3) livres de posicionamento. Um resumo dos artigos citados é mostrado na Tabela 1.

Na primeira categoria, os protocolos requerem uma estimativa da localização global dos nós (latitude/longitude). Tal estimativa pode ser obtida equipando-se todos os nós com receptores GPS (*Global Positioning System*), através da execução de um algoritmo de localização ou mesmo através do posicionamento manual dos nós [Boukerche et al. 2007]. Nesta categoria, um dos trabalhos mais relevantes da literatura sobre protocolos de roteamento geográfico é o *Greedy Perimeter Stateless Routing (GPSR)* [Karp and Kung 2000], que utiliza a posição dos nós e o destino do pacote para tomar decisões de encaminhamento guloso, através das informações sobre os vizinhos a um salto de distância. Sua utilização consiste de dois métodos de encaminhamento de pacotes: encaminhamento guloso, usado quando aplicável, e encaminhamento por perímetro, utilizado em regiões onde o primeiro falha. Neste último método, o algoritmo é capaz de lidar com a ocorrência de mínimos locais. O *Bounded Voronoi Greedy Forwarding (BVGF)* [Xing et al. 2004], é um algoritmo de localização que realiza decisões de encaminhamento guloso baseadas na localização dos vizinhos a um salto de distância. Neste algoritmo, uma rede é modelada através de um diagrama de Voronoi onde as regiões representam as localizações dos sensores. Cada nó conhece sua posição geográfica e mantém uma tabela com os endereços dos nós vizinhos e, para manter esta tabela, cada nó divulga periodicamente, uma mensagem de *broadcast* que inclui sua localização, bem como a localizações dos vértices das regiões de Voronoi. Já no algoritmo *Energy Aware Greedy Routing (EAGR)* [Haider et al. 2007], cada nó conhece sua localização geográfica e a de seus vizinhos, seu nível de energia e de seus vizinhos e executa uma decisão local para a escolha do próximo salto não somente baseado na localização dos nós, mas também nos seus níveis de energia, de modo a aumentar o tempo de vida da rede como um todo.

Tabela 1. Comparação dos algoritmos de roteamento geográficos.

	A	**B**	**C**	**D**	**E**	**F**
BVGF [Xing et al. 2004], [Xing 2006]		√				
EAGR [Haider et al. 2007]		√		√		
GF-RSSI [Pham et al. 2006]	√	√	√			
GF-ViP [You et al. 2009]	√					
GMFP [Panigrahi et al. 2009]	√			√		
GPSR [Karp and Kung 2000]		√				√
GRIP [Zhang et al. 2015]	√					√
RSSR [Boukerche et al. 2008]	√	√	√			
RSSR with hole bypass capability [Oliveira et al. 2015]	√	√	√			√
RADR [Lima et al. 2013]	√	√	√	√	√	
ARESTA	√	√	√	√	√	√

A) Não requer coordenadas globais.
B) Não requer coordenadas virtuais.
C) Utiliza RSSI como métrica
D) Ciente de energia.
E) Realiza agregação de dados.
F) Identifica buracos de roteamento.

Os algoritmos baseados em coordenadas virtuais estão na segunda categoria. Tais algoritmos tentam criar um outro tipo de sistema de coordenadas, não relacionado a um sistema de posicionamento global, onde a informação de localização não está disponível [Huang et al. 2011]. Neste contexto, o *Greedy Minimum energy consumption Forwarding Protocol (GMFP)* [Panigrahi et al. 2009] foi desenvolvido com o objetivo de aumentar o tempo de vida da rede, através da escolha ótima de um nó elegível em cada salto, utilizando encaminhamento geográfico guloso. O algoritmo *Greedy Forwarding with Virtual Position (GF-ViP)* [You et al. 2009], utiliza um esquema de encaminhamento guloso baseado no progresso do encaminhamento em direção ao *sink*. O algoritmo foi proposto com dois complementos, o *Virtual Multi-Level Position (MVP)* e o *Greedy Forwarding with Virtual Hierarchical Position (HVP)*. Cada algoritmo possui duas variações que utilizam algoritmos Gulosos baseados em progresso. A principal vantagem nestas abordagens é a aplicação do encaminhamento guloso durante todo o processo de roteamento, gerando resultados altamente eficientes na geração de rotas. Ainda nesta categoria, o *Greedy Routing through dIstributed Parametrization for guaranteed delivery in WSNs (GRIP)* proposto em [Zhang et al. 2015], utiliza a técnica de encaminhamento guloso para prover garantia de entrega através da parametrização do domínio de rede real em um domínio virtual. A abordagem utiliza um algoritmo de detecção de perímetro para identificar nós na borda da rede e aproveita essa informação para atribuir coordenadas iniciais.

Finalmente, na terceira categoria, os nós não precisam conhecer suas informações de localização nem as de seus vizinhos mas utilizam alguma outra informação para indicar a direção que o pacote deve seguir. Neste cenário, o *Greedy Forward with Received Signal Strength Indicator (GF-RSSI)* [Pham et al. 2006] utiliza a potência do RSSI recebido como um filtro usado para gerar rotas. No *Received Signal Strength Routing (RSSR)*, proposto em [Boukerche et al. 2008], o *sink* também é equipado com um dispositivo de comunicação de maior potência e seu pacote de consulta atinge todos os nós da rede em um único salto. O pacote de resposta dos nós sensores é encaminhado para o vizinho que recebeu a mensagem do *sink* com a maior potência que, na teoria, é o vizinho mais próximo do *sink*. Uma melhoria que permite ao algoritmo RSSR ser capaz de detectar e desviar buracos foi apresenta em [Oliveira et al. 2015]. Uma abordagem de encaminhamento guloso que explora a grande capacidade de comunicação do nó *sink* foi proposta em [Lima et al. 2013]. Neste trabalho, o algoritmo RADR utiliza uma técnica de encaminhamento guloso que é combinada aos valores obtidos pelo RSSI do *sink* e dos nós regulares para implementar roteamento geográfico e agregação de dados.

Nossa proposta atual, o ARESTA, difere das soluções citadas pois não requer a utilização de sistemas de coordenadas globais ou virtuais. Além disso, é capaz de lidar com buracos de roteamento durante o processo de encaminhamento das informações ao *sink*, mantendo todas as vantagens de um algoritmo de roteamento geográfico mas sem a necessidade de conhecer as informações sobre o posicionamento dos nós. Nosso algoritmo, ainda realiza agregação de dados utilizando para isso os valores de RSSI dos nós para estimar as distâncias entre vizinhos e o nó *sink*. Neste contexto, a agregação de dados foi utilizada para reduzir o tráfego de pacotes na rede e aumentar a vida útil dos nós.

3. ARESTA - Agregação de Dados com Desvio de Buracos para Redes de Sensores sem Fio com *Sinks* de Alto Alcance

Neste trabalho está sendo proposto um algoritmo de roteamento geográfico e agregação de dados, que utiliza um *sink* com grande capacidade de comunicação e é capaz de lidar com de buracos de roteamento chamado ARESTA (Agregação de Dados com desvio de buracos para Redes de Sensores sem Fio com *Sinks* de alto alcance). O algoritmo foi desenvolvido para ter todos os benefícios de um algoritmo de roteamento geográfico, como confiabilidade, escalabilidade e altas taxas de entrega, sem a necessidade da informação de posicionamento dos nós sensores, além de executar agregação de dados durante o processo de encaminhamento dos pacotes ao *sink*. Nosso algoritmo ainda é capaz de descobrir e desviar buracos de roteamento em RSSFs, através da construção de uma rota alternativa construída a partir de nossa regra de desvio de buracos executada até que um vizinho próximo ao destino seja encontrado.

Nossa solução considera a possibilidade de se equipar o nó *sink* com um dispositivo de comunicação de alto alcance de forma que em um único salto o nó *sink* pode enviar uma consulta para toda a rede, aproveitando o grande alcance na comunicação. O aspecto principal do algoritmo ARESTA é ser capaz detectar e desviar buracos de roteamento, traçando rotas alternativas até encontrar um vizinho que esteja geograficamente próximo ao destino, em nosso caso o *sink*. A motivação para o desenvolvimento desta melhoria surgiu a partir do projeto ATTO (*Amazonian Tall Tower Observatory*), que já está implementado na Amazônia e consiste em uma torre no meio da floresta com uma estrutura física de 320 m de altura e alta capacidade de comunicação. O objetivo do projeto é mapear eventos na região e fornecer medidas confiáveis de fontes e sumidouros de gases de efeito estufa como CO_2, CH_4 e N_2O [Tollefson 2010].

Na arquitetura proposta, a mesma mensagem de consulta do nó *sink* alcançará os nós da rede com diferentes potências (RSSIs) de tal forma que os nós mais distantes possuirão um sinal mais baixo, ao contrário dos nós mais próximos, que irão receber a mensagem com um sinal mais forte, conforme ilustrado na Figura 1. A confiabilidade do RSSI, utilizado como métrica, é discutida em [Holland et al. 2006, Jacinto 2012].

A proposta, descrita no Algoritmo 1, inicia quando o nó *sink* envia uma consulta para toda a rede (linha 10). Os nós sensores, ao receberem essa consulta, estimam suas distâncias ao nó *sink* usando a técnica de RSSI (linhas 11 e 12), conforme ilustrado na Figura 2(a). Em seguida, cada nó irá enviar a seus vizinhos um pacote de anúncio, que contém a sua distância estimada para o *sink* (linha 13). Ao mesmo tempo, os nós receberão diversos pacotes deste tipo, um de cada vizinho, e salvarão tais dados de distâncias nas suas tabelas de roteamento (linha 19 e 20). Cada nó irá verificar se possui dados a serem retornados à consulta do *sink*. Se tiver, tais dados serão adicionados à lista de dados agregados (linhas 14–16). O próximo passo do algoritmo é calcular um tempo de espera para agregação dos dados antes de enviar a resposta ao *sink*. Um temporizador é iniciado com o tempo de espera calculado (linhas 17 e 18). O tempo para o envio dos pacotes é definido de acordo com a potência do RSSI recebido por cada nó, ou seja, quanto mais baixo o sinal do RSSI, menos tempo o nó terá que esperar para enviar seus pacotes em direção ao *sink*. Após a etapa de configuração, as informações agregadas a serem encaminhadas ao próximo salto são atualizadas (linha 21).

Antes do temporizador de um nó expirar, o nó corrente checa se há um vizinho

Algorithm 1 Algoritmo ARESTA

Variáveis:
1: $tabelaVizinhos_i = \emptyset$; {Distâncias dos vizinhos ao *sink*}
2: $dadosAgregados_i = \emptyset$; {Conjunto de dados agregados}
3: $dadosConsulta_i$; {Dados obtidos a partir da consulta do *sink*}
4: $tempoEnvio_i$; {Tempo de espera para envio da resposta}
5: $temporizador_i$; {Temporizador usado na agregação de dados}
6: $proximoSalto_i$; {Próximo salto para o envio dos dados agregados}
7: $saltoElegivel_i$; {Próximo salto elegível em caso de desvio de buraco}
8: $dist_i$; {Distância para o *sink*}

Evento:
9: *Sink* recebe da Central de Monitoramento uma $consulta(consultaId_k, consulta_k)$;

Ação:
10: Envia $consulta(consultaId_k, consulta_k)$ para todos os nós.

Evento:
11: $msg_i = consulta(consulta_k, consulta_k)$;
12: $dist_i = estimaDistancia(msg_i)$;
Ação:
13: Envia $anuncio(n_i, dist_i)\ \forall\ n_j \in vizinho_i$. {Divulgando a distância}
14: **Se** $dadosConsulta_i = recRetornoConsulta(consulta_k)$ **Então**
15: $dadosAgregados_i := dadosAgregados_i \cup dadosConsulta_i$; {Agrega informação ao pacote}
16: **Fim Se**
17: $tempoEnvio_i = estimaTempoEnvio(dist_i)$; {Tempo de agregação}
18: $temporizador_i.inicia(tempoEnvio_i)$; {Inicia temporizador}

Evento:
19: $msg_i = anuncio(vizinho_k, dist_k)$;
20: $tabelaVizinhos_i = tabelaVizinhos_i \cup (vizinho_k, dist_k)$;
21: $proximoSalto_i = vizinhoProximoAoSink()$;
Ação:
22: **Se** $dist_i < proximoSalto_i.dist$ **Então** {Estou em um buraco de roteamento?}
23: $saltoElegivel_i := desvioDeBuraco(tabelaVizinhos_i)$; {Qualifica o próximo salto como elegível}
24: Envia $resposta(dadosAgregados_i)$ para $saltoElegivel_i$;
25: **Fim Se**

Evento:
26: $temporizador_i.expirou()$; {Tempo para agregação expirou}
Ação:
27: Envia $resposta(dadosAgregados_i)$ para $proximoSalto_i$; {Encaminha dado agregado}
28: $dadosAgregados_i = \emptyset$;

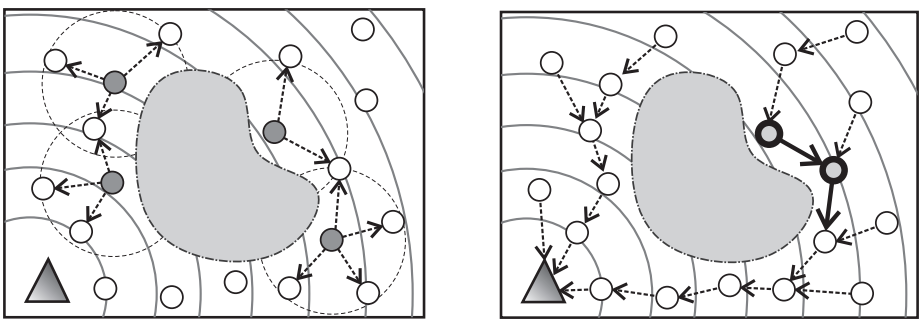
Evento:
29: $msg_i = resposta(dadosAgregados_k)$;
Ação:
30: **Se** $n_i \neq 0$ **Então** {Sou um nó regular?}
31: $dadosAgregados_i := dadosAgregados \cup dadosAgregados_k$;
32: **Se** $temporizador_i.inativo()$ **Então** {O dado foi enviado?}
33: Envia $resposta(dadosAgregados)$ para $proximoSalto_i$;
34: $dadosAgregados_i = \emptyset$;
35: **Fim Se**
36: **Senão** {Eu sou o nó *Sink*}
37: Envia o dado recebido à Central de Monitoramento.
38: **Fim Se**

mais próximo ao *sink*. Caso contrário, isto significa que o nó está em uma região onde a utilização da técnica de encaminhamento guloso não é possível. O nó corrente então executa regra de detecção e contorno de buracos (linha 23), que seleciona um novo vizinho em direção ao *sink* a partir da tabela, ilustrado na Figura 2(b). A esquema de deteção de contorno de buracos permite que o próximo salto seja eleito de acordo com a distância do *sink* em relação a cada vizinho, considerando como elegível o vizinho geograficamente mais próximo do *sink* em relação ao nó corrente.

O nó da região de buracos não possui vizinhos elegíveis para o encaminhamento de pacotes através da técnica de encaminhamento guloso. Neste sentido, o encaminhamento de um pacote contornando o buraco (linha 24) é feito de acordo com as distâncias dos vizinhos armazenadas na tabela de roteamento de cada nó, de forma que o vizinho a ser escolhido como próximo salto esteja o mais distante possível em relação ao nó corrente, e o mais próximo possível em relação ao *sink*. Esta mesma mensagem contém o identificador de cada mensagem encaminhada ao nó que se encontra uma região de buracos. Tais informações serão utilizadas para atualizar a tabela de roteamento dos nós que permite a escolha do próximo salto. Este processo é repetido até que seja escolhido um vizinho apto a executar o encaminhamento guloso.

Quando o temporizador de um nó expira (linha 26), um pacote com todos os dados agregados é enviado ao vizinho mais próximo do nó *sink* (linha 27). Este pacote será recebido pelo vizinho escolhido (linha 29) que, provavelmente, não terá seu temporizador expirado ainda, por estar mais próximo do *sink*. Este último nó irá simplesmente agregar os dados recebidos em sua lista de dados agregados (linha 32) e continuar esperando que seu temporizador expire para enviar esses dados adiante (linhas 26–28). Finalmente, quando o *sink* recebe algum pacote de resposta, ele envia os dados recebidos de volta à central de monitoramento (linhas 37).

Em alguns casos raros, é possível que o temporizador de um determinado nó tenha expirado (e o seu dado agregado já tenha sido encaminhado) antes de receber todos os dados dos seus vizinhos. Isso pode ocorrer na presença de erros de RSSI elevados, propagados durante o processo estimativa de distância e configuração do temporizador para a agregação de dados. Caso isto ocorra, o pacote recebido é encaminhado imediatamente ao vizinho mais próximo do *sink*, através da técnica de encaminhamento guloso (linhas 32–35). Nesta proposta, utilizamos a agregação de dados com forma de reduzir o número de informações redundantes e aumentar o tempo de vida da rede, também proporcionado

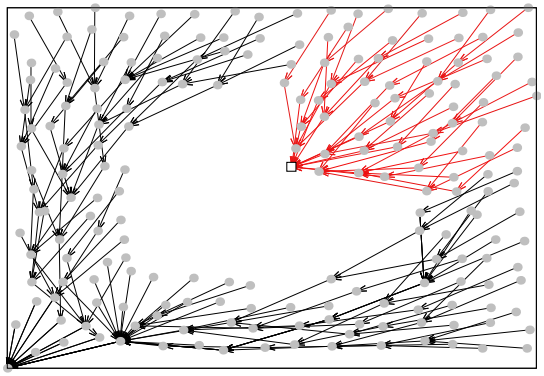


(a) (b)

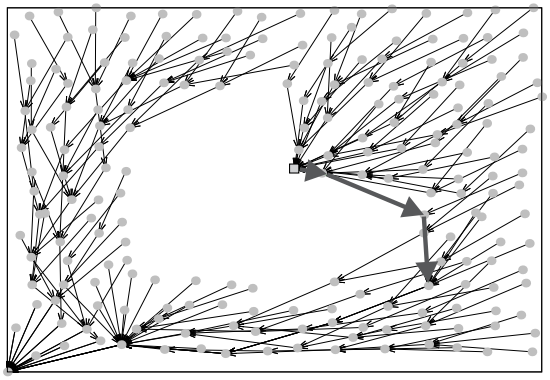
Figura 2. ARESTA: (a) Após a mensagem de anúncio do RSSI, cada nó estima sua distância em relação ao *Sink*. (b) Regra de desvio de buraco sendo aplicada.

pela economia de energia que é um benefício direto da agregação de dados.

Com o objetivo de visualizar e comparar as rotas geradas pelos algoritmos, a Figura 3(a) apresenta um gráfico de fluxo com as rotas geradas pelo algoritmo RADR, que não possui desvio de buracos, enquanto a Figura 3(b) apresenta as rotas geradas pelo algoritmo ARESTA. Os gráficos foram gerados a partir de uma simulação realizada e é possível ver claramente o fluxo com o escoamento dos pacotes formado pela abordagem proposta no cenário com a presença de um buraco (ao centro).



(a) Algoritmo RADR: nó não consegue identificar um buraco.



(b) Algoritmo ARESTA: nó detecta um buraco de roteamento e aplica a regra de desvio.

Figura 3. RADR e ARESTA: gráfico de fluxo de pacotes. *Sink* localizado no canto inferior esquerdo.

4. Avaliação de Performance

Na presente seção, avaliamos o algoritmo ARESTA proposto e comparamos sua performance com o algoritmo RADR apresentado em [Lima et al. 2013].

4.1. Metodologia

A avaliação de performance foi realizada através de simulações usando o simulador Sinalgo [ETH-Zurich 2014], que provê um ambiente completo para simulação de algoritmos distribuídos. As simulações foram realizadas em um campo de sensoreamento de $130\,\text{m} \times 130\,\text{m}^2$. O posicionamento dos nós foi feito com base em uma grade perturbada, mantendo a densidade de $0.03\,\text{nós/m}^2$. Em todos os resultados, as curvas representam valores médios, enquanto as barras de erro representam intervalos de confiança para 95 % de confiança a partir de 33 instâncias independentes (sementes aleatórias).

Tabela 2. Configuração do cenário de simulação.

Parâmetro	Valor padrão
Área monitorada	$130\,\text{m} \times 130\,\text{m}^2$
Quantidade de nós	512 nós (grade perturbada)
Densidade	$0.03\,\text{nós/m}^2$
Alcance da comunicação	20 m
Imprecisão do RSSI a curta distância	15 % da dist. real
Posição do Sink	Canto inferior esquerdo

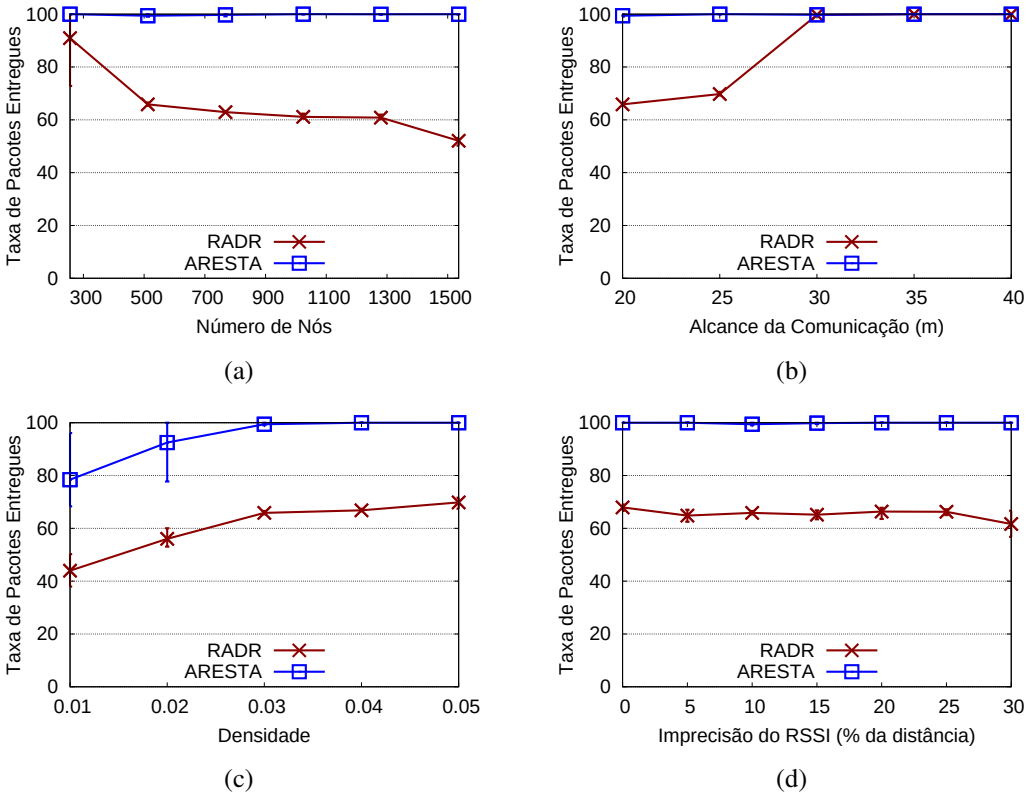


Figura 4. Resultados obtidos ao se avaliar a taxa de entrega.

O nó *sink* foi posicionado no canto inferior esquerdo enquanto que os nós regulares ocupam a grade uniformemente. Também introduzimos um um buraco de roteamento na rede, conforme apresentado na Figura 2, configurado de acordo o alcance máximo dos nós vizinhos. Neste cenário, os nós em uma extremidade do buraco não podem se comunicar com nós da outra extremidade. Os parâmetros de simulação foram baseados no sensor MicaZ e os valores utilizados são mostrados na Tabela 2.

4.2. Impacto da Escalabilidade da Rede

A escalabilidade de rede foi avaliada através do aumento do número de nós na rede de 256 para 1536 nós, aumentando-se o tamanho da área monitorada de modo a manter a densidade constante. Na Figura 4(a), podemos observar que o algoritmo RADR não consegue entregar todos os dados, uma vez que não é capaz de lidar com buracos de roteamento. Por outro lado, o algoritmo ARESTA consegue entregar todos os pacotes na maioria das vezes. O resultado demonstra claramente que nossa solução é capaz de operar na presença de buracos, até mesmo quando aumentamos o número de nós.

4.3. Impacto do Raio de Comunicação

Para avaliarmos o impacto do raio de comunicação dos nós em nosso algoritmo, aumentamos este parâmetro de 20 m até 40 m. Raios de comunicação pequenos reduzem o número de nós vizinhos, o que consequentemente reduz o número de possibilidades para o próximo salto [Boukerche et al. 2008]. A Figura 4(b), mostra que o número de pacotes entregues com sucesso do algoritmo RADR diminui para cerca de 65 % quando o raio de comunicação é 20 m e fica em torno de 70 % quando o alcance aumenta para 25 m.

Neste caso, os nós possuem poucos vizinhos elegíveis, gerando ainda mais buracos de rede. Neste cenário, o algoritmo RADR interrompe o processo de encaminhamento guloso. Contudo, em todos os casos nosso algoritmo ARESTA é capaz de entregar todos os dados agregados.

4.4. Impacto da Densidade da Rede

O impacto da densidade da rede em nosso algoritmo foi avaliado começando com uma densidade relativamente baixa de 0.01 nós/m^2 e aumentando tal densidade até 0.05 nós/m^2. Em densidades baixas, um nó terá poucos vizinhos, dificultando a agregação e consequentemente o encaminhamento do pacote adiante, enquanto que ao se aumentar a densidade, o número de vizinhos de cada nó aumentará. Podemos observar pela Figura 4(c) que o algoritmo ARESTA continua funcionando corretamente e é capaz de operar de forma eficiente mesmo em baixas densidades ao contrário do algoritmo RADR, que em baixas densidades consegue entregar cerca de 40 % dos pacotes. Por fim, observamos que número de pacotes entregues com sucesso pelo algoritmo ARESTA é de cerca de 80 % quando a densidade está em 0.01 nós/m^2. Neste caso, os nós possuem poucos vizinhos e isso aumenta as chances de escolha de um salto elegível para o encaminhamento dos pacotes.

4.5. Impacto da Imprecisão do RSSI a curta distância

Neste trabalho avaliamos o impacto da imprecisão do RSSI a curta distância (estimativa de distância entre os nós regulares) aumentando essa imprecisão de 0 % para 30 % da distância real entre os nós. Como é possível observar na Figura 4(d), o algoritmo ARESTA não é afetado de forma significativa pela imprecisão do RSSI, além de se mostrar confiável no encaminhamento de pacotes ao *sink*. Por outro lado, o algoritmo RADR é afetado durante o processo de estimativa de distância entre os vizinhos prejudicando as decisões de próximo salto a partir de uma área com a presença de um buraco.

4.6. Impacto da Regra de Desvio de Buracos na escolha do próximo salto.

O impacto da solução de desvio de buracos em relação à escolha do próximo salto também foi avaliado. Neste sentido, comparamos os resultados obtidos através da execução dos algoritmos RADR e ARESTA na presença de um buraco de roteamento. A grade foi configurada com a inserção de um polígono côncavo irregular em forma de *L*, ao centro conforme Figura 2, de forma que os nós de um lado do buraco não se comunicassem com os nós do outro lado, portanto a parte oposta do buraco. Esta configuração pode ser considerada como o pior cenário, já que um buraco baseado em uma circunferência não cria um buraco de roteamento nem mesmo denota o fenômeno dos mínimos locais. Outros tipos de polígonos não podem criar cenários de pior caso para o problema explorado neste trabalho.

O número de escolhas erradas em relação ao próximo salto no algoritmo ARESTA é um pouco maior do que no algoritmo RADR com poucos nós, como mostram as Figuras 5(a), 5(b), 5(c) e 5(d), pois no desvio de buracos os pacotes são encaminhados a vizinhos que não estão geograficamente mais próximos do *sink* e sim na borda do buraco. Apesar disto, o algoritmo ARESTA é capaz de gerar árvores de escoamento eficientes enquanto em direção *sink*, contornando a borda do buraco como já mostrado na Figura 3(b).

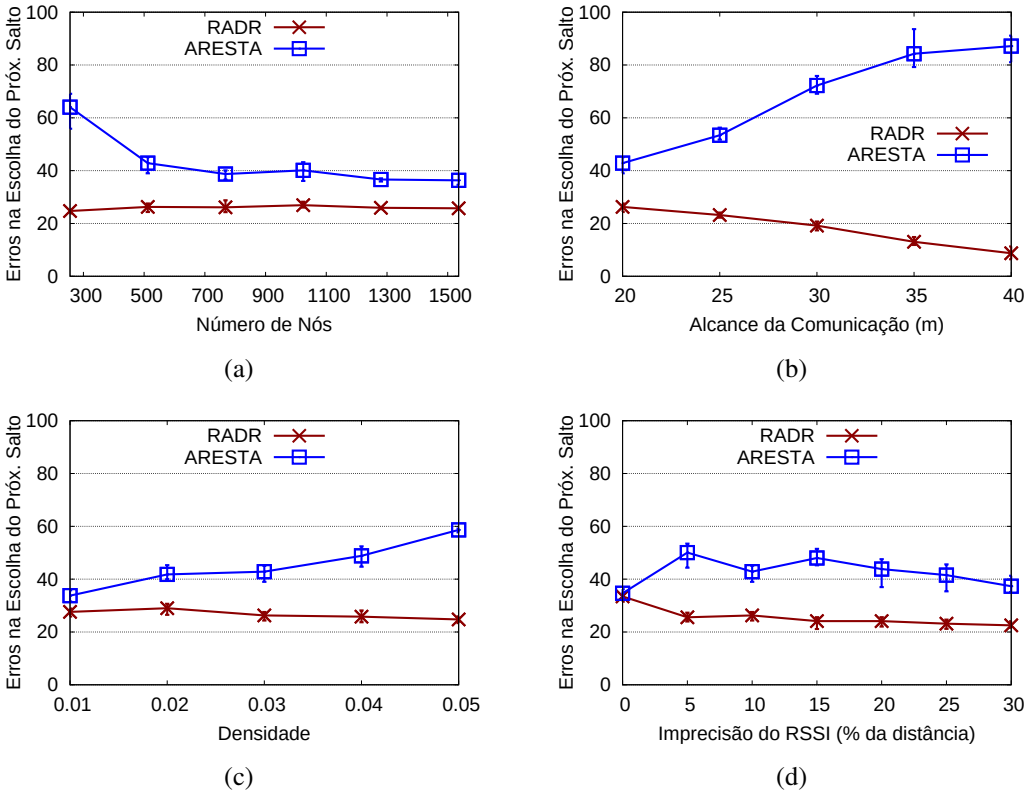


Figura 5. Resultados obtidos ao se avaliar escolha do próximo salto.

5. Aplicabilidade da Solução Proposta

Em cenários de Redes de Sensores em Fio, alguns obstáculos naturais como lagos, construções ou regiões sem a presença de sensores podem causa a interrupção do encaminhamento de pacotes. Em algoritmos de roteamento geográfico, a existência de uma área em que os nós regulares não conseguem encaminhar adiante seus pacotes de dados é problema dos mínimos locais, amplamente estudado em [Choi and Choo 2011, Chang et al. 2012]. Neste trabalho consideramos a possibilidade de se equipar o nó *sink* com um dispositivo de comunicação de maior potência, de forma que em um único salto toda a rede seja alcança por uma consulta, também única do *sink*. Esta é uma suposição razoável em cenários como o do projeto ATTO (Amazonian Tall Tower Observatory), em que pretendemos utilizar e melhorar esta arquitetura de comunicação. Neste contexto, o algoritmo ARESTA pode ser utilizado como uma solução de roteamento guloso para o encaminhamento de pacotes, a partir de nós sensores implantados ao redor da torre. Por exemplo, caso hajam obstáculos naturais no ambiente sensoreado, o ARESTA estaria apto a detectar e aplicar a regra de desvio. Neste sentido, o algoritmo ARESTA proporciona um aumento do tempo de vida dos nós que participam da agregação de dados.

6. Conclusão

Neste trabalho propomos um novo algoritmo de roteamento geográfico com agregação de dados para redes de sensores, que chamamos de ARESTA (Agregação de Dados com desvio de buracos para Redes de Sensores sem Fio com *Sinks* de alto alcance). O aspecto essencial desta abordagem é equipar o nó *sink* com um potente dispositivo de comunicação de forma que em um único um pacote de anúncio todos os nós da rede sejam alcançados.

Com base apenas nos valores de RSSI obtidos pelos nós, propomos uma nova e eficiente abordagem o roteamento e agregação de dados, com a capacidade de lidar com buracos de roteamento.

O algoritmo ARESTA, é capaz de detectar e superar buracos em uma RSSF através da construção de uma rota alternativa até o vizinho mais próximo do *sink*. A agregação de dados ocorre durante o processo de encaminhamento dos pacotes e é possível graças à utilização de um temporizador configurado em cada nó regular após a consulta inicial recebida pelo nó *sink*. Uma série de experimentos foi realizada para avaliar a solução proposta. Os resultados demonstram claramente os benefícios introduzidos pela técnica de desvio de buracos, somada a abordagem simples de encaminhamento guloso e agregação de dados.

Referências

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002a). A survey on sensor networks. *IEEE, Communications Magazine*, 40:102–114.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002b). Wireless sensor networks: a survey. *Computer Networks*, 38:393–422.

Al Alawi, R. (2011). Rssi based location estimation in wireless sensors networks. In *17th IEEE International Conference on Networks (ICON), 2011*, pages 118–122.

Boukerche, A., Oliveira, H., Nakamura, E., and Loureiro, A. (2007). Localization Systems for Wireless Sensor Networks. *IEEE Wireless Communications, 2007*, 14(6):6 –12.

Boukerche, A., Oliveira, H., Nakamura, E., and Loureiro, A. (2008). A Novel Location-Free Greedy Forward Algorithm for Wireless Sensor Networks. In *IEEE International Conference on Communications (ICC), 2008*.

Chang, C.-Y., Hung, L.-L., Wang, T.-C., and Wang, T.-L. (2012). Orzbp: An obstacle-resistant zone-based broadcasting protocol for wireless sensor networks. In *14th International Conference on High Performance Computing and Communication, 2012*, pages 714–721.

Chen, D. and Varshney, P. (2007). A survey of void handling techniques for geographic routing in wireless networks. *Communications Surveys Tutorials, IEEE*, 9(1):50–67.

Choi, M. and Choo, H. (2011). Bypassing hole scheme using observer packets for geographic routing in wsns. In *Information Networking (ICOIN), 2011 International Conference on*, pages 435–440.

Éfren Souza, Nakamura, E., Oliveira, H., and Figueiredo, C. M. S. (2012). Reducing the impact of location errors for target tracking in wireless sensor networks. *Journal of the Brazilian Computer Society*.

ETH-Zurich, D. C. G. a. (2014). Sinalgo - simulator for network algorithms. In *http://dcg.ethz.ch/projects/sinalgo/*.

Haider, R., Javed, M., and Khattak, N. (2007). Eagr: Energy aware greedy routing in sensor networks. In *Future Generation Communication and Networking (FGCN 2007)*, volume 2, page 344349.

Holland, M., Aures, R., and Heinzelman, W. (2006). Experimental investigation of radio performance in wireless sensor networks. In *Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on*, pages 140 –150.

Huang, P., Wang, C., and Xiao, L. (2011). Improving end-to-end routing performance of greedy forwarding in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, PP(99):1.

Jacinto, R. M. P. (2012). Modelação da Propagação numa Rede de Sensores sem Fios. Master's thesis, Universidade Nova de Lisboa.

Karp, B. and Kung, H. T. (2000). Gpsr: Greedy perimeter stateless routing for wireless networks. In *MOBICOM*, pages 243–254.

Kim, Y.-J., Govindan, R., Karp, B., and Shenker, S. (2005). Geographic routing made practical. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 217–230, Berkeley, CA, USA. USENIX Association.

Lima, M., Oliveira, H., Nakamura, E., and Loureiro, A. (2013). Roteamento e Agregação de Dados baseado no RSSI em Redes de Sensores sem Fio. In 31^o *Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos (SBRC 2013)*, Brasilia, DF.

Oliveira, H. A., Boukerche, A., Guidoni, D. L., Nakamura, E. F., Mini, R. A., and Loureiro, A. A. (2015). An enhanced location-free greedy forward algorithm with hole bypass capability in wireless sensor networks. *J. Parallel Distrib. Comput.*, 77(C):1–10.

Panigrahi, B., De, S., and Sun Luk, J.-D. (2009). A greedy minimum energy consumption forwarding protocol for wireless sensor networks. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1 –6.

Pham, N., Youn, J., and Won, C. (2006). A comparison of wireless sensor network routing protocols on an experimental testbed. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006*, volume 2, pages 276 –281.

Tollefson, J. (2010). A towering experiment an ambitious project to track greenhouse gases from a perch high above the amazon forest will provide crucial data - but only if scientists can get it built. *Nature*, 467(386).

Xing, G. (2006). *Unified power management in wireless sensor networks*. PhD thesis, St. Louis, MO, USA. AAI3238702.

Xing, G., Lu, C., Pless, R., and Huang, Q. (2004). On greedy geographic routing algorithms in sensing-covered networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '04, pages 31–42, New York, NY, USA. ACM.

You, J., Lieckfeldt, D., Han, Q., Salzmann, J., and Timmermann, D. (2009). Look-ahead geographic routing for sensor networks. In *IEEE International Conference on Pervasive Computing and Communications (PerCom), 2009*, pages 1 –6.

Zhang, M., Li, F., He, Y., Lin, J., Gu, X., and Luo, J. (2015). GRIP: Greedy Routing through dIstributed Parametrization for guaranteed delivery in WSNs. *Springer Wireless Networks*, 21(1):67–80.

A Distributed Coverage Node Scheduling Algorithm for Dense Wireless Sensor Networks

Daniel R. Matos[1], **Gabriel A. L. Paillard**[2], **Miguel F. de Castro**[1]

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)
Campus do Pici, bloco 910 – 60.440-900 – Fortaleza – CE – Brazil

[2]Instituto Universidade Virtual – Universidade Federal do Ceará (UFC)
Av. Humberto Monte, s/n, bloco 901, Campus do Pici
60.440-554 – Fortaleza – CE – Brazil

daniel@dc.ufc.br, gabriel@virtual.ufc.br, miguel@dc.ufc.br

Abstract. *Wireless Sensor Networks (WSNs) have remarkable application potentials: smart homes, environment monitoring, health care, and a myriad of other commercial areas. In general, WSNs operate under severe energy constraints: a typical sensor's battery is irreplaceable and its lifetime is limited. Moreover, those tiny sensor nodes usually have extremely limited physical capabilities as well. Consequently, sensor deployments with reasonable densities are fundamental in order to prolong the lifetime of the networks. Distributing sensors randomly can lead to a redundancy on some areas and this is desirable to overcome the failures of some sensors. In this work, we propose a distributed algorithm to schedule active sensors to both reduce the redundancy of data obtained by the network and prolong its lifetime. Experimental evaluation shows that our approach guarantees a coverage rate around* 99.9%.

1. Introduction

Advances in electronic devices with the introduction of new miniaturization techniques, lower power consumption and increasing processing power, is making the widespread use of devices featuring appealing power and wireless communication capabilities feasible. Wireless Sensor Networks make extensive use of this kind of devices for a variety of applications, such as environmental monitoring, agriculture, health monitoring, education, climate monitoring, and military uses, among others. In addition to its traditional applications, the so-called Internet of Things (IoT) poses a significant demand to WSN, since there is a set of technologies that provide connectivity at any moment and in all locations for IoT components [Atzori et al. 2010].

Every year, electronic devices gains more processor power. The computational capacity of a huge computer in the 60's can now be easily exceeded by a simple small calculator. Despite this evolution in the computational power, battery capacity did not evolve as quickly. Innovation in energy storage area has not kept pace with the growing demand of electronic devices, making it even more important to use the available computing resources efficiently. Moreover, even with the increase of available processing at an increasingly lower cost, it is still necessary to use cheaper devices with higher energy constraints, e.g., sensors to be embedded in the walls of a building to prevent its collapse or to assist in the search for survivors of disasters.

One of the most used techniques for prolonging network lifetime is coverage management, wherein coverage can be defined in terms of a percentage of the monitored space that is covered by the available sensors. This technique allows the turn off of the devices that are in the same sensor area, thereby reducing redundancy and, consequently, increasing the network's lifetime.

WSN usually assumes that the cost of sensors is low, such that a large amount of sensors can be used to monitor an area, specially when the sensors are randomly disposed to avoid black holes, i.e., no coverage areas. Therefore, coverage management turns sensors on and off in order to save resources, reducing redundancy in the gathered data. Solutions developed in this area must consider high level objectives like robustness, scalability and simplicity [Wang and Xiao 2006].

There are a lot of approaches in energy-efficient scheduling mechanisms, each of those considering different design assumptions, such as detection model, sensing area, transmission range, and different target applications [Wang and Xiao 2006]. This work focuses on a simple and elegant solution that can be used in virtually any WSN scenario that allows scheduling. We employed the Scheduling by Edge Reversal (SER) [Barbosa and Gafni 1989] to handle the active nodes in a WSN. The use of this algorithm, originally developed to distributed systems to enter critical region, gives rise to challenges that are unveiled and covered later in this paper. The main design goals of this work is to conceive a simple, scalable and robust solution.

The remainder of this article is organized as follows. Section 2 presents the state-of-the-art on WSN coverage. Scheduling by edge reversal is introduced in section 3. Our approach is detailed in section 4. Section 5 is devoted to present the simulations we designed for experimental evaluation and we analyze the results obtained. Finally, the conclusions of this work and further research perspectives are shown in section 6.

2. Coverage in Wireless Sensor Networks

The coverage in a WSN indicate how a target area is covered by the deployed sensors. There are some coverage models in the literature [Wang 2011]. Most of them use geometric relations to check whether a point is covered by a specific sensor. Boolean models have just one possible result for a given point and a target area: covered (TRUE) or not covered (FALSE). Such models are classified as boolean sector coverage models, that consider a circular sector, i.e., a boolean disk coverage model. In this case, each sensor has a radius and all points inside that circle are covered by that sensor.

There are other network coverage models that consider other factors. The Attenuated Disk Model considers that coverage quality may decrease depending on the distance between the target point and the closest sensor. Therefore, this model defines an attenuation function that models such behavior. The Truncated Disk Model is similar to the Attenuated Disk Model, but imposes a limit to the distance from a sensor to a target. Hence, this model can be understood as a mix of Boolean Disk Coverage Model and Attenuated Disk Model. Moreover, there are also stochastic coverage models built on probabilistic frameworks.

According to [Wang 2011], the main design issues for coverage problems are: *coverage type, deployment method, coverage degree, coverage ratio, activity scheduling and network connectivity*. Briefly speaking, each one of these are:

- **Coverage Type** refers to what kind of target will be covered: *Point Coverage* considers the targets as discrete points inside a monitored area, hence the main goal is to cover all points; *Area Coverage* treats all points inside the monitored area equally, whereas the main objective is to cover the entire area; and *Barrier Coverage*, where the objective is to create a barrier and find a penetration path inside this barrier to the covered targets.
- **Deployment method** refers to how the sensors will be deployed in the target area. Sensors can be placed deterministically in specific points, or can be randomly placed, such as being dropped from an air-plane.
- **Coverage Degree** refers to how a point is covered by WSN. This characteristic shows how many sensors are covering a specific point, when a points is *k-covered*, there are *k* sensors covering that point.
- **Coverage Rate** is a value that indicates how much of target area is monitored (or how many points).
- **Node Scheduling** changes the states of the sensors to active or inactive. If a point is covered by more than one sensor, the scheduling method decides which sensors can be active and for which period time. This is the main area of this work.
- **Network Connectivity** guarantees that all pairs of nodes in the network can be reached, i.e., there is always a path between any pair of nodes.

2.1. Routing in WSN

The routing of collected information in a WSN poses a major challenge in this field. A typical WSN has hundreds or thousands of nodes such that it is not possible to keep a global addressing system, an agent responsible for routing or an address table maintenance. Sensors must act together, in a distributed way, in order to assure that sensed data will be within the reach of the base station.

There are several routing protocols for WSN in the literature, which can be divided according to their network structure in three types: plain routing, hierarchical routing and location based routing [Al-Karaki and Kamal 2004]. In plain routing, all sensors have the same rules and functions, these may belong to a particular routing path. In hierarchical routing, sensors are grouped into clusters and have different roles on the network, each cluster has a member exercising the role of a cluster head, a central node that receives all information from its components and sends to the base station. In location-based routing sensors, positions are available (the nodes are placed in a deterministic way or have some positioning system, like GPS) and this information is used for routing decisions.

3. Scheduling by Edge Reversal

Consider a neighborhood-constrained system composed of a set of *processing elements* (PEs) and a set of *atomic shared resources* represented by a connected directed graph $G = (V, E)$, where V is the set of PEs and E is the set of its directed edges (or arcs), stating the access topology (directed edges are henceforth referred to as arcs). The latter is defined in the following way: an arc exists between any two nodes *if and only if* the two corresponding PEs share at least one atomic resource. Scheduling by Edge Reversal (SER) works as follows: starting from any acyclic orientation ω on G, there is at least one *sink* node, i.e., a node such that all its arcs are directed to itself; all sink nodes are allowed to operate while other nodes remain idle.

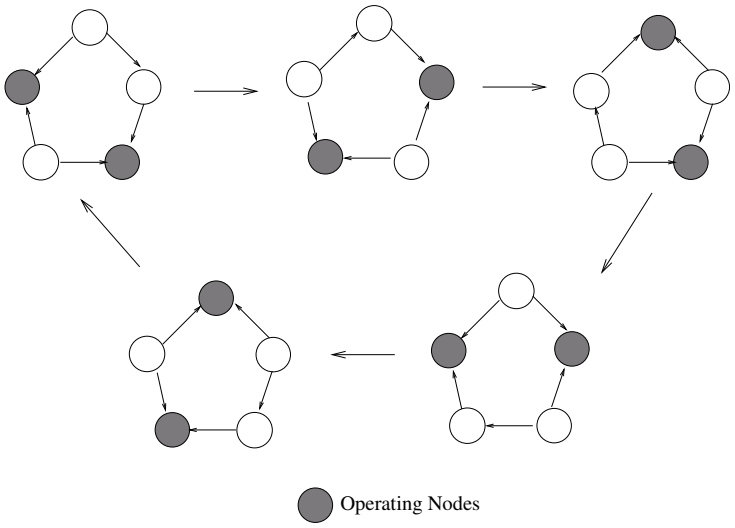


Figure 1. SER dynamics for the Dining Philosophers under heavy load

This obviously ensures mutual exclusion at any access made to shared resources by sink nodes. After operation, a sink node will reverse the orientation of its arcs, becoming a *source* and thus releasing the access to resources to its neighbors. A new acyclic orientation is defined and the whole process is then repeated for the new set of sinks. Let $\tilde{\omega} = g(\omega)$ denote this greedy operation. SER can be regarded as the endless repetition of the application of $g(\omega)$ upon G.

Assuming that G is finite, it is easy to see that eventually a set of acyclic orientations will be repeated defining a period of length P. This simple dynamics ensures that no deadlocks or starvation will ever occur since in every acyclic orientation there exists at least one sink, i.e., one node allowed to operate. In addition, it can be proved that in any period, every node operates exactly the same constant number of times (denoted M) [Barbosa and Gafni 1989].

SER is a fully distributed graph dynamics in which the sense of time is defined by its own operation, i.e., the synchronous behavior is equivalent to the case where every node in G takes an identical amount of time to operate and also an identical amount of time to reverse arcs. Another interesting observation to be made here is that any topology G will have its own set of possible SER dynamics [Barbosa and Gafni 1989].

As an example of SER's applicability, consider Dijkstra's paradigmatic Dining Philosophers problem under heavy load, i.e., in the case philosophers are either "hungry" or "eating" (no "thinking" state). Such system can be represented by a set $\{P_1, \ldots, P_N\}$ of N PEs, in which each PE shares a resource both with its previous PE and its subsequent PE. Thus, taking the original configuration where $N = 5$ and setting an acyclic orientation over the 5 nodes ring, the resulting SER dynamics where $P = 5$ and $M = 2$ is illustrated in Fig. 1.

4. Distributed Scheduling Proposal

The motivation for this work came across the evidence of the possibility of using the scheduling algorithm by reversal edges (SER) to scale the radio nodes in a wireless sensor network and model this solution to solve active nodes scheduling in a coverage manage-

ment problem. The Scheduling by Edge Reversal (SER) algorithm was the chosen one and this solution was used to schedule radios in WSN in order to maintain the coverage of the sensed area and reduce energy consumption. It is worth mentioning that the radio of a sensor can consume more energy receiving messages than transmitting, as we can observe in a data-sheet of Texas Instruments CC2420 radio [Texas 2013]. This algorithm made use of message exchange for coordinate active sensor scheduling, and turning off the radio is by itself a big challenge, because messages cannot be be received when the radio has been turned off.

4.1. Initializing

The first pass was mapping the SER algorithm to schedule radio sensors in a WSN. Our neighborhood-constrained system is the WSN itself. The *processing elements* are the sensors nodes in WSN. In a general way, this can be done as follows: if two sensors in a WSN are close enough, we can consider it redundant, so one of them can be turned off (so this is the clause to consider that the two PE's have an *atomic shared resource* in SER). Then, if two sensors are redundant, there will be an edge between them. After mapping redundant sensors and creating edges between them, the SER algorithm can be applied. In order to maintain the SER correctness, we assume that the nodes have a unique identifier and that the edges are initially set in direction to higher id sensors, as this is a requirement for avoid deadlocks in the use of the SER algorithm.

The first pass of our proposal is to identify all neighbors of a sensor. For this, all nodes send an initial *hello* message and wait for messages from its neighbors. With these incoming messages, the nodes can make an estimation of distances, based on the radio transmission. As this is not the focus of this work and we do not want to use euclidean distances (considering that all sensor knows its position), we used RSSI to estimate these distances. RSSI is not reliable to calculate distances [Heurtefeux and Valois 2012] [Benkic et al. 2008], as it can vary too much in greater distances. Here we just create an edge if nodes are close enough to have considerable redundancy in the sensed area, so the precision in this small distance is sufficient for this work.

In our algorithm we consider only the messages exchanged between sensors close enough, for this we use a reduced power to save energy in message exchange.

4.2. The Lonely Sensor Case

As SER was not originally intented to solve schedule in WSN, we identified a situation that can generate black holes in coverage, specially in network borders. We call these situations *lonely sensors case*. This particular case occurs when a node has only one neighbor and this neighbor, in turn, has more than a neighbor. Figure 2 shows an example. In (a), node 7 has all its edges directed for itself, becoming active. In (b) node 7 reverses its edges, making nodes 1 and 3 active. In (c) nodes 1 and 3 reverse their edges, enabling node 2, at this point node 1 could still remain active, because there are no neighbors active - node 7 will only become active when receiving the edge of 2 next round. This may cause a hole in the monitored area since nodes 1 and 7 will not be active.

In this case, the node that has only one neighbor could continue active after finishing its active time, once its neighbor gets the edge, but still have the edges of other neighbors, ie, the node being off will cause the appearance of a hole in the network that could be avoided if the application needs the largest possible coverage.

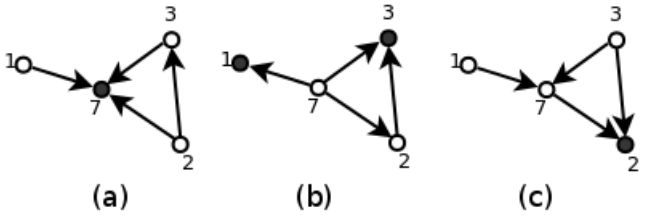


Figure 2. ***In the original SER when a node has only one neighbor, it may occur that both this node and its neighbor are disabled at the same time, which is not a problem in distributed systems, but could cause a hole in the coverage on WSN***

To prevent this situation from occurring, when all sensors identified their neighbors, they broadcast their neighbor list. In this way, all nodes know their neighbors and all neighbors of each one, making possible to identify when a node is in *alone sensor case*. When this happens, a sensor in *lonely sensor case* will not reverse this edge after the end of the round, and it will remain active until one of its neighbors reverts the mentioned edge. When a node is in this condition, it has all edges directed to itself, except the one of the lonely sensor, and it will take this one, finally scheduling that node to sleep.

The first difference in SER usage is in identifying lonely sensors, the inserted amendment aimed to maintain the node active up until its neighbor get its adjacent edges, which maintains the underlying acyclic graph (avoiding ”deadlocks”) and the alternation between nodes accessing their shared resources remains guaranteed (avoiding ”starvation”). The last point is the analysis of the residual energy of the sensors, which may be a delay in one or more cycles in the reversal of the edges and this can result in unequal access to shared resources but the correctness of the new algorithm remains.

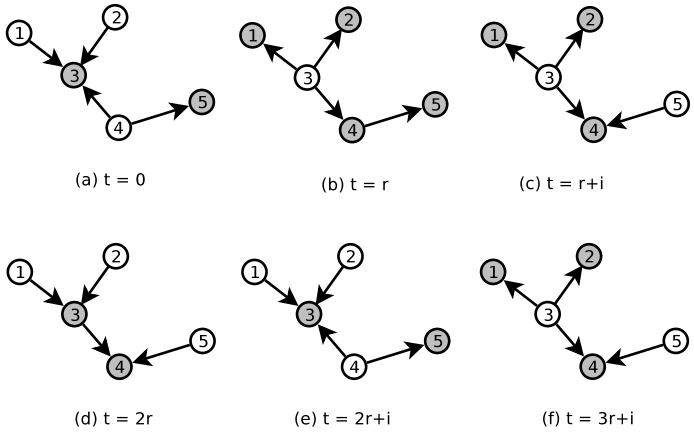


Figure 3. When there is a delay, as in (c) where the node 5 reversed its edge with delay i. In some rounds, the nodes would be again synchronized, because this time will be propagated to the rest of the network.

4.3. Synchronization

If a sensor tries to send a message to a neighbor that has a radio turned off, the message will get lost. In order to avoid that, one solution is the use of synchronized time in all nodes, which it is not always possible. We employ a scheme with the use of timers, counting round times. It works as follows: when a sensor inverts his edges, before putting

itself in a sleep state, a timer is started with a value of a round. When this timer expires, the sensor turns on its radio allowing it to receive messages from its neighbors again. After all incoming messages in that round have been received, it starts a new timer, but now the value of this timer is the duration of that round plus the elapsed time until the receiving of the last message from this round. In this way, we can guarantee that the sensor will be awake at the same moment of the first node, which ended the previous round, avoiding the loss of incoming messages from sensors using distinct timers. With this solution, a delay in a node clock would not be a problem, except for the waste of energy, provided that the delay is not in the order of magnitude of the time of a round, which could cause that the delayed node was considered dead. In some rounds the delay would be propagated to all nodes, as shown in Figure 3 which illustrates the synchronization scheme employed.

4.4. Energy Level and Edge Reversal

Another change in the original SER algorithm consists in the use of current energy level of a sensor to decide if it will or not invert its edges in current round. This strategy will balance the energy levels, allowing an energy consumption in a more elaborate way. Using this strategy creates a problem because of the expectation that the algorithm will reverse all edges in each round, and a sensor cannot be in an active state for an entire round waiting a message that will not come. In order to avoid this, when a round ends, an active sensor checks if this remaining energy is bigger than their neighbors energy level, if so, it will remain active for one more round. We added a field in all communications messages containing actual energy level of the sensors. When an active sensor checks its energy comparing with its neighbors, it will be checking the energy of the last round, but all neighbors remain in sleep state in this round, so not much energy change is expected. When a sensor decides not to invert, it needs to send a message announcing that it will not proceed to invert its edges in this round, so its neighbors can sleep for one more round.

When a sensor receives a message of this kind (not invert), it will check if there are still other edges to receive in this round, if not, it can go to sleep state and notify this to its neighbors. Using energy level to decide edge reversal can impact coverage causing black holes, because a sensor will break the edge reversal cycle. To solve this problem, when a sensor realizes that all its neighbors will be in a sleep state, it changes to an active state, avoiding the black hole creation. This is why a node needs to send a message when it is going to a sleep state, when the expected is to be in an active state.

Using the energy level as a decision before the reversal strategy of edges can affect the network coverage. We have in the left side of Figure 4 the inversion of edges without taking into account the energy level, at each iteration, the active nodes reverse its edges (simple use of scheduling by edge reversal). Note that in 1 - (d) is a repetition of the cycle which continues to occur continuously every three iterations. In 2 we have the inversion edge taking into account that we can not reverse its edges due to comparing the energy level with neighbors. 2 - (c) node 3 decided not reverse its edge by having more energy than neighbors. So during that round he was the one node to stay active. Coverage during the iteration 2 - (c) is reduced due to the reversal cycle break.

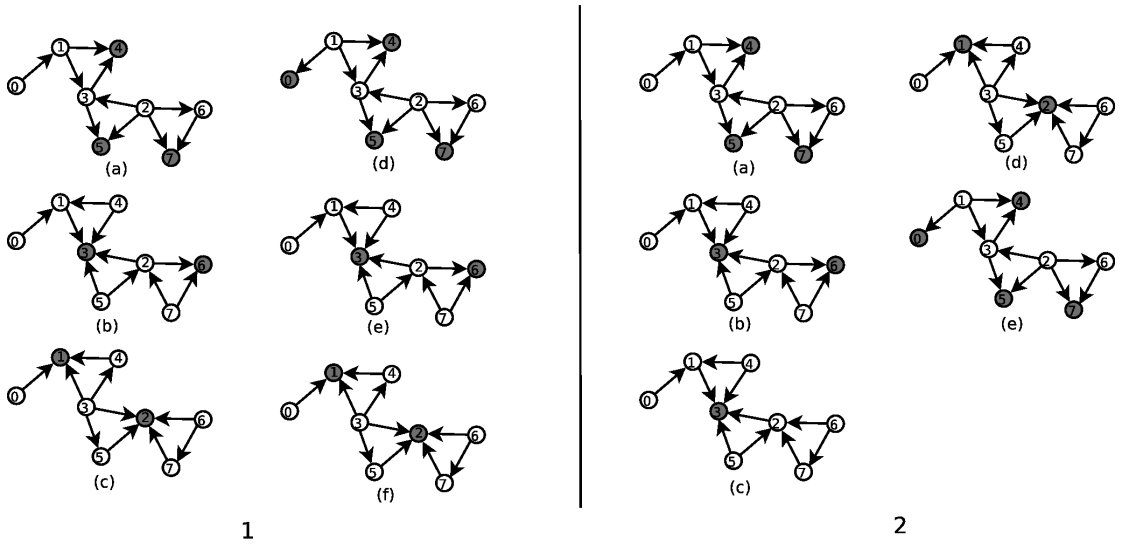


Figure 4. Inversion edges without taking into account the amount of energy (1) Inversion edges taking into account the amount of energy (2)

4.5. A Distributed Coverage Node Scheduling Algorithm for Dense Wireless Sensor Networks

The proposed solution is showed right below in high level pseudo code. Some details, related to the startup of the solution which includes the use of a probabilistic algorithm for the exchange of identities between neighboring nodes, have been omitted but they are detailed in [Paillard et al. 2004].

$Find\ neighborhood$
$Ajust\ Edges$; `Orient edges to higher ID nodes`
$Verify\ Edges$; `Check if all edges point to itself`
if $ActiveState$ **then**
  $StartTimer(EndRound\ :\ RoundTime)$;
else
  $Go\ to\ Sleep$
  $StartTimer(EndRound\ :\ RoundTime)$; `section 4.3`
end

Algorithm 1: Distributed Duty Sheculing

Algorithm 1, which is the main algorithm of the solution, starts by sending initial messages and waiting messages from its neighborhood. Shortly thereafter, another message send the neighborhood list, as stated in section 4.2. From there it checks if it has all the edges, getting active to the end of the round, or whether it should go to the inactive state, if it doesn't have all the edges. Fig 5 shows a diagram state for a better understanding of our entire approach.

Algorithm 2 is executed when one of the timers expires. At the end of a round, the node checks the level of energy in relation to its neighbors, reverses the edges and turns to an inactive state, or send a message annoucing that it will continue in the active state. When a node finishes a round in which it was in an inactive state, it turns on the radio and waits for messages from neighbors also starting the timer that will expire only if there are

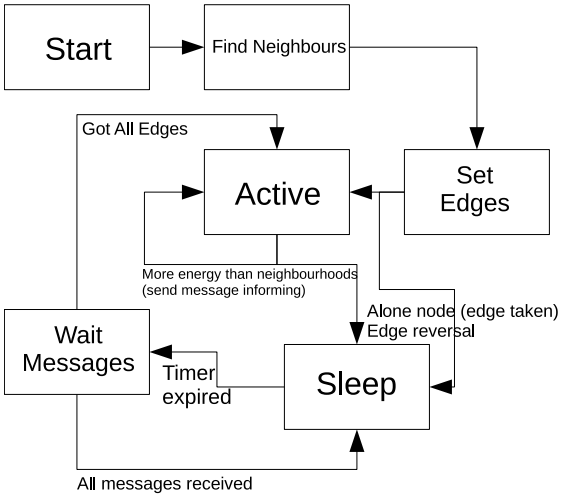


Figure 5. State Diagram

missing messages from any node at the end of the round - this will cause the removal of this node, considering him dead.

Apart from these we have two algorithms that operate when a node receives a message, verifying and executing the steps taken when receiving each type of message, such as whether the node will remain active or not [Matos 2013].

5. Results and analysis

Below are presented the obtained results by simulating the work proposed here and a discussion of results.

5.1. Simulator and parameters

The simulations were performed on the Castalia Simulator [Castalia 2013], version 3.2. The choice of the simulator was based mainly on the following facts: Castalia is WSN specific, open source and is widely utilized [Pediaditakis et al. 2010]. Castalia was developed on *Omnet++* framework [Varga 2001, Varga and Hornig 2008].

We employed an hierarchical architecture where cluster heads were arranged in a grid, for a total of 16 nodes arranged in $4\ x\ 4$ grid equally distributed inside a $100m\ x\ 100m$ simulation area. The routing to the nodes was defined statically following the shortest path to the sink (located at center). As routing is not our focus, we used this schema because of its ease to implement. Cluster Heads send one message when the simulation starts and each sensor associates a cluster head with more powerful signal. The application layer used just generates one package per second in each sensor, simulating sensor readings. The MAC utilized was TunnableMAC (a CSMA/CA implementation available in simulator).

Boolean disk model was used to calculate coverage rate in each round of the simulation (as a percent of total area), since Castalia does not provide a direct output of coverage and boolean disk is a model widely used in the literature.

We considered that an edge will exist between two sensors if the distance measured between them has a maximum of 6 meters. Measures on simulator with RSSI

```
begin
    switch ExpiredTimer do
        case EndRound
            if I don't have more energy than my neighbors then
                SendMsgReverseEdges() ;          Edge Reversal
                Turn off the radio;
                TimerStart(EndIdleTime : (RoundTime));
            else
                sends message that will not reverse
                SendMsgNotReverse();
                TimerStart(EndRoundWithTime : (RoundTime));
            end
        end
        case EndIdleTime
            Activates the radio again and waits for new
            messages
            Turn on the Radio;
            Activate neighbors check timer
            TimerStart CheckNodes with time : RoundTime;
        end
        case CheckNodes
            A neighbor doesn't send messages during a
            round find which one and remove
            LookneighborOff
        end
    endsw
end
```

Algorithm 2: Expired Timer

started showing inconsistency for distances higher than 8 meters, so, up to this distance, there are no problems in measurement estimation method utilized. We simulated a communication with -3dbm signal and obtained 24.84 meters of medium range (repeated 30 times). With this value we considered a sensing range of 12 meters (about half of communication range, as no measurement was less than 24 meters), what give about 62% of common area.

Sensors were randomly deployed and simulations were executed with the number of sensor ranging from 100 to 250 (we did not consider Cluster Heads), each being repeated 30 times. The total time of simulation was 500 seconds.

We executed three different configurations, as follows:

1. ECNS: Energy-Efficient and Coverage-specific Node Scheduling for Wireless Sensor Networks [Meng et al. 2010]. This solution was chosen because of its similarities in being a fully distributed, location aware and simple solution;
2. Hierarchical: No active sensor scheduling, all nodes send their data to Cluster Heads all time;
3. Our Proposal: Distributed Duty Scheduling Algorithm for Large Scale WSN's.

The solution proposed on this work, using adapted version of SER applied to WSN's.

5.2. Round time

To set how long will take a round, we made simulations starting from $5s$ to $75s$, corresponding to 1% to 15%, respectively, of the total amount of time execution, allowing at least six rounds to execute. We observed that with larger round times, more energy will be left at the end, but this is because some nodes have not yet been scheduled, so as $5s$ was the round time that resulted in a more balanced energy distribution between nodes, we chose this time to run the remaining simulations.

5.3. Coverage Rate Results

As the algorithm will schedule sensors, coverage rate is an important measure to verify the viability of our solution. Calculate the area covered by sensors in a WSN is not an easy job[Aziz et al. 2009, Huang and Tseng 2003] and Castalia simulator does not this output available, so we need to create an output in each round and calculate overlapping areas using boolean sensor model. As expected, with more dense networks, less impact on coverage rate will occur. Table 1 shows the results.

Table 1. Coverage Rate

Number of Sensors	Coverage obtainned
100	90%
150	98%
200	99.75%
250	99.9%

As ECNS requires coverage rate as input, it was not directly compared here. We used the coverage rate computed by our algorithm as input for the ECNS to calculate energy levels.

5.4. Energy Consumption

Initial energy in each node was set to 90 Joules (about 1% of capacity of an AA battery). This value was chosen to reduce simulation time necessary until the first sensor dies. The energy results showed that our solution can save energy, every in low density simulations, but with lost of coverage, as seen in later section (90% coverage with 100 nodes). Fig 6 show results for 100 and 250 sensors respectively. ECNS starts saving energy with 250 nodes, but our proposal saved 20% more energy than ECNS and 35% compared to do not schedule sensors in this situation.

5.5. Energy Distribution

To verify as was the energy balance of our solution, we show the results of the spatial distribution of the energy level at the end of a simulation as we can observe in Fig 7. The hierarchical graph is virtually a flat, since all nodes were active all the time. On ECNs, as few of us have been disabled, there is also little energy gap, so we chose to override the results in the same image. More above you can see that there is a variation in energy distribution, since some nodes performed more rounds than others, but the depressions and ridges are small, showing a good balance in consumption between the nodes.

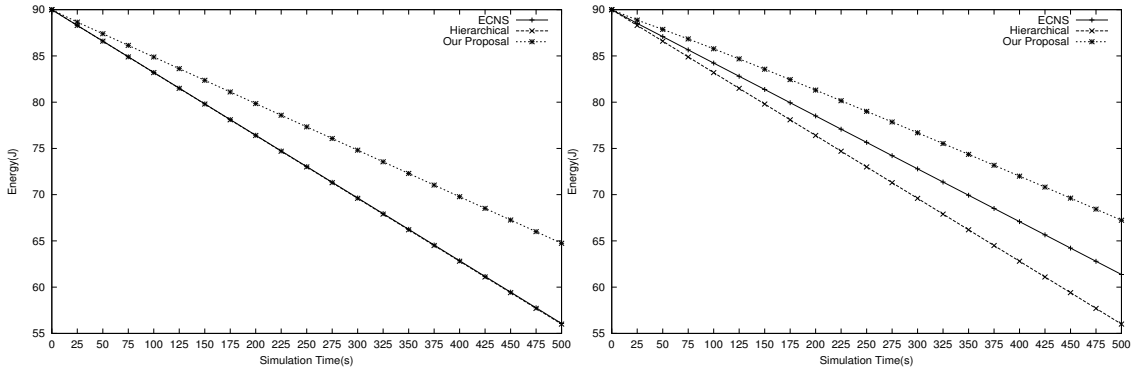


Figure 6. medium residual energy for 100 and 250 sensors

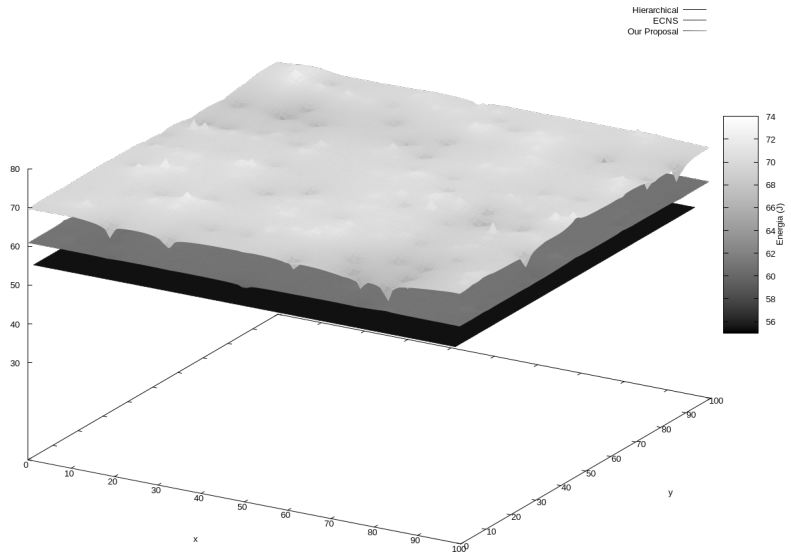


Figure 7. Spatial Energy distribution.

5.6. Network Lifetime

Network lifetime is the time an important feature in a WSN, since one of the main objectives is to extend the network lifetime as much as possible. The definition of Network lifetime may vary depending on the application since there are applications that work correctly only if they have data of all network sensors, can work with already other data from at least a proportion of functioning sensors. Network lifetime was defined here as the time until death of the first sensor (the battery of sensor was completely drained). Fig 8 shows the results in minutes (remembering that the amount of initial energy of the nodes is greatly reduced in our simulations, so the results in minutes).

6. Conclusions

As shown by the results, our solution provides good energy saving with low impact on coverage in dense networks and enhances network lifetime. If coverage required is not so high, the solution can be even used in low density networks. The ECNS is simple, based on message exchanges between neighbors during execution rounds, similar to the solution

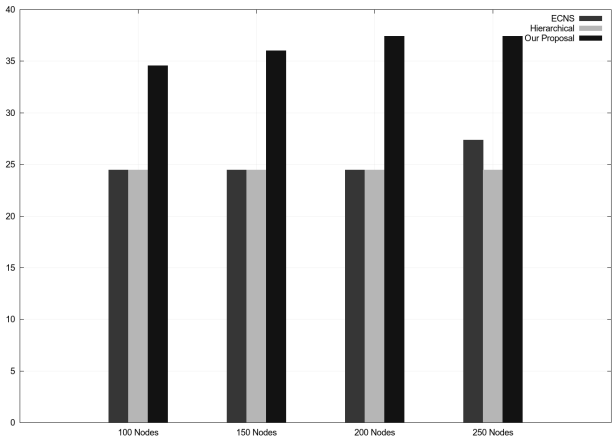


Figure 8. Medium time until first sensor die (in minutes).

proposed here. We have not found other specific solutions based on these same principles. ECNS was compared to two other solutions: LDAS and NRS regarding QoC. Our analysis focuses on energy conservation and the results matched the coverage obtained with the ECNS with much less active nodes.

ECNS starts showing good results only on high dense network, what is expected since it puts a node in inactive state if it has a high number of neighbors in order to guarantee coverage (the exact number depends on coverage rate required). We did not test with higher densities because we obtained 99.9% of coverage rate using 250 sensors.

This work can be easily adapted to heterogeneous networks (with different sensing radio) changing the calculation of distances, taking into account different radio.

The distance between sensors to enable an edge can have an important influence on results of solution. Considering higher distances will lead to more energy saving, but it will decrease coverage rate. Thus, this is application dependant.

As a future work, we could have an integrated solution with routing that can lead to more energy saving. Another future work will be to improve the algorithm to calculate the distances between nodes, as RSSI is not suitable in real world applications. It is possible to develop dynamic intervals between edge reversal, with higher or lower times, depending on network state.

References

Al-Karaki, J. and Kamal, A. (2004). Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *The International Journal of Computer and Telecommunications Networking*, 54(15):2787–2805.

Aziz, N. A. A., Aziz, K. A., and Ismail, W. Z. W. (2009). Coverage strategies for wireless sensor networks. 3(2):134 – 140.

Barbosa, V. C. and Gafni, E. (1989). Concurrency in heavily loaded neighborhood-constrained systems. *ACM Trans. Program. Lang. Syst.*, 11(4):562–584.

Benkic, K., Malajner, M., Planinsic, P., and Cucej, Z. (2008). Using rssi value for distance estimation in wireless sensor networks based on zigbee. In *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*, pages 303–306.

Castalia (2013). Castalia wireless network simulator.

Heurtefeux, K. and Valois, F. (2012). Is rssi a good choice for localization in wireless sensor network? In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 732–739.

Huang, C.-F. and Tseng, Y.-C. (2003). The coverage problem in a wireless sensor network. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, WSNA '03, pages 115–121, New York, NY, USA. ACM.

Matos, D. R. (2013). Um algoritmo distribuído para escalonamento de sensores em rssf. Master's thesis, Universidade Federal do Ceará, Departamento de Computação, Programa de Pós-Graduação em Computação, Fortaleza - Ceará.

Meng, F., Wang, H., Wei, G., and Fan, Z. (2010). Energy-efficient and coverage-specific node scheduling for wireless sensor networks. In *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*, MSWIM '10, pages 368–375, New York, NY, USA. ACM.

Paillard, G., Djerourou, F., Lavault, C., and Ravelomanana, V. (2004). Assigning codes in a random wireless network. In Souza, J. N., Dini, P., and Lorenz, P., editors, *Telecommunications and Networking - ICT 2004*, volume 3124 of *Lecture Notes in Computer Science*, pages 348–353. Springer Berlin Heidelberg.

Pediaditakis, D., Tselishchev, Y., and Boulis, A. (2010). Performance and scalability evaluation of the castalia wireless sensor network simulator. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 53:1–53:6, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Texas (2013). Cc2420(active) single-chip 2.4 ghz ieee 802.15.4 compliant and zigbee ready rf transceiver.

Varga, A. (2001). The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference (ESM'2001)*.

Varga, A. and Hornig, R. (2008). An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Wang, B. (2011). Coverage problems in sensor networks: A survey. *ACM Comput. Surv.*, 43(4):32:1–32:53.

Wang, L. and Xiao, Y. (2006). A survey of energy-efficient scheduling mechanisms in sensor networks. *Mob. Netw. Appl.*, 11(5):723–740.

ALABAMO: A LoAd BAlancing MOdel for RPL

Tarcisio Bruno Oliveira[1], **Pedro Henrique Gomes**[2]
Danielo G. Gomes[1], **Bhaskar Krishnamachari**[2]

[1]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Departamento de Engenharia de Teleinformática – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brazil

[2]Autonomous Networks Research Group (ANRG)
Ming Hsieh Department of Electrical Engineering – University of Southern California (USC)
Los Angeles – CA – USA

{tarcisiocarneiro,dgomes}@great.ufc.br

{pdasilva,bkrishna}@usc.edu

Abstract. *Most of the nodes that compose the Internet of Things (IoT) are battery-operated, which makes energy efficiency a critical goal. However, more important than simply saving energy is consuming energy uniformly among the nodes. Imbalanced energy consumption may disrupt the network (energy hole problem). We present a solution that solves the network load balance problem; it is based on the RPL protocol, making it suitable for dynamic environments and compliant with standard network stack for IoT devices. The proposal was implemented in ContikiOS and tested in a real 41-node testbed using TelosB motes. Network lifetime had a 2-fold increase compared to the default RPL implementation. We also reduced the standard deviation of consumption by 50.64%, which indicates that nodes spend energy homogeneously, thus extending the lifetime of most of the nodes.*

1. Introduction

The Low-Power and Lossy Networks (LLN) are comprised of several sensing embedded systems that communicate and enable different applications, such as smart building, habitat monitoring [Mainwaring et al. 2002] and industrial automation. These networks are part of the Internet of Things (IoT), which will be the biggest driver for productivity growth [Accenture 2014]. The Internet of Things (IoT) is expected to have tens of billions of devices by the end of the decade.

In most of the LLN-based applications, the sensor nodes have constrained resources: memory, processing power and energy. Considering these limited capabilities, the routing protocols need to be well designed, since protocols impact the use of node's resources and energy consumption. In addition to providing reliable and efficient delivery, the routing algorithms need to take care of energy efficiency in order to extend the network lifetime.

In 2012, the IETF ROLL working group published the RFC 6550, which specifies a Routing Protocol for Low-Power and Lossy Networks (RPL). The RPL became the standardized IPv6-compliant routing protocol for LLNs. It is a distance vector protocol that establishes Destination Oriented Directed Acyclic Graphs (DODAGs) based on links

and/or nodes metrics [Winter et al. 2012]. A DODAG is a tree-like topology that supports both downstream and upstream traffic and is built according to an Objective Function (OF).

Different Objective Functions can be designed in order to achieve specific optimization criteria and satisfy the requirements of a particular application. Supplementing RPL specification, the ROLL working group also standardized the Minimum Rank with Hysteresis Objective Function (MHROF). MHROF is designed to be extended with different metrics to calculate the cost of a path. A default metric defined by RFC 6719, which is used in most of RPL implementations, uses the Expected Transmission Count (ETX) of links to calculate the path with minimum cost. With DODAGs created using MHROF with the ETX metric, nodes tend to select parents with best link quality and smallest hop-count to the root node.

Even though selecting the parent with best link quality is a local optimum choice, it may cause a load-balancing problem. In scenarios where nodes distribution and/or traffic pattern are heterogeneous, the workload imbalance becomes critical. Nodes that are closer to the root and/or that have many children with good-quality links will receive and forward a large number of packets which may quickly deplete their battery life.

To tackle the above issues, we propose a new Objective Function based on MHROF. The new OF uses both the traffic profile of the nodes and the ETX of the links in order to solve the imbalance problem. The proposal is called **ALABAMO** (**A L**o**A**d **BA**lancing **MO**del for RPL) and it is a flexible model that uses two input parameters (workload characterization) and improves the task of choosing the preferred parent in RPL aiming at the extension of the network lifetime.

In this context, the main contribution of this paper is a new Objective Function (implemented in ALABAMO) that is compatible with RPL and provides traffic-aware balanced routing. The main target application of ALABAMO is multi-hop data collection applications, such as envrionmental monitoring. We implemented ALABAMO in a real operating system and evaluated it in a testbed with 41 nodes. Afterwards, we compared our solution with the default MHROF.

The paper is organized as follows. Section 2 provides the background of RPL and its issues. The ALABAMO model and its algorithm is described in Section 3. Following, Section 4 evaluates the performance of our proposal. We present prior work in Section 5. Finally, Section 6 concludes this paper.

2. Background and Problem Statement

Routing protocols for LLNs have to face challenging problems such as unreliable wireless links, lack of infrastructure and constrained resources. RPL was developed to work efficiently in networks comprised of thousand of nodes with limited resources and high packet loss. It is an adaptable routing protocol that dissociates packet processing and forwarding from the routing optimization. Decoupled from its routing core, the Objective Function (OF) is the way of achieving the optimization criteria required by the applications, such as minimum energy consumption or minimum end-to-end latency. An OF describes how nodes should convert one or more metrics and/or constraints into a rank. A rank represents the node's distance to the Directed Acyclic Graph (DAG) root. In ad-

dition, an OF also is used to assist nodes in choosing the best parent to be used in the upwards direction to reach the root.

2.1. RPL Overview

RPL builds a DAG that is composed of one or more Destination Oriented DAGs (DODAGs), each one rooted at a border router or sink node. RPL supports all three types of traffic: multipoint-to-point, point-to-multipoint and point-to-point traffic [Clausen et al. 2014]. Four types of custom ICMPv6 control messages are used to build and maintain the routing topology [Conta et al. 2006]:

- DIO (DODAG Information Object) : Message that carry information about the DODAG, such as RPLInstanceID, Version Number, Rank and the routing metrics used to compute the routes. The messages are transmitted through the network to construct and maintain the DODAG. Only the root node can start their dissemination. Every node that joined a DODAG also broadcasts DIO periodically;
- DIS (DODAG Information Solicitation): A node can send a DIS message to one (unicast) or multiple (multicast) neighbors, to proactively solicit configuration information;
- DAO (Destination Advertisement Object): These messages are used when downward traffic is needed;
- DAO-ACK (Destination Advertisement Object Acknowledgement): This control message reveals if the neighbor that received the DAO intends to be a previous hop for the sender in the downward route.

2.1.1. Topology Formation

The DODAG construction starts when the root node broadcasts a DIO message that indicates the initial configuration parameters. The root's neighbors take a rank based on the objective function and choose their preferred parent. The node's ranks are at least *MinHopRankIncrease* larger than the root's rank. Following, the root's neighbors start to broadcast their own DIO messages. As the process converges, each node in the network receives DIO messages and has a preferred parent set.

RPL implements the Trickle Algorithm [Levis et al. 2011] to reduce the rate at which the DIO messages are sent after the topology has reached an stable point. This mechanism prevents the nodes from wasting energy, since it continually doubles the transmission period until it reaches a maximum value. However, when some inconsistency is detected, e.g. a loop, link disruption or parent change, Trickle resets the timeout to a minimum value in order to promptly propagate the updated DODAG.

2.1.2. Objective Function (OF)

RPL's DODAG construction process is mainly determined by the Objective Function (OF) and the path metrics. An OF defines how a node computes its rank, which is based on one or more metrics. A rank represents the relative distance to the DODAG root. Moreover, it also describes how a node must choose its preferred parent. Thus far, the IETF ROLL

working group detailed two OFs, which are implemented in most of the operating systems developed for LLNs, such as TinyOS and ContikiOS:

- Objective Function Zero (OF0): This OF uses hop-count as a routing metric. A node calculates its rank by adding a positive and indirectly normalized scalar value to its preferred parent rank;
- Minimum Rank With Hysteresis Objective Function (MHROF): This OF chooses routes that minimize additive routing metrics such as latency, hop-count and ETX [Vasseur et al. 2011]. In addition, MHROF uses hysteresis to reduce instability due to small metric changes. ContikiOS uses MHROF with Expected Transmission Count (ETX) routing metric as default objective function.

2.2. RPL Problem

Both objective functions proposed by the IETF ROLL group aim to reduce the number of packet retransmissions. While OF0 tries to minimize the number of hop-count, MHROF focus on the pairwise communication quality, based on the ETX metric.

In a network using RPL with MHROF based on ETX metric, nodes will choose the parents with best link quality. Considering a network with non-uniform distribution and uneven data traffic, it may result in significant load imbalance. The sensor nodes that have links with better quality will forward more packet than the others. Thereby, their energy depletion will be notably faster than the nodes with light traffic, and thus, gaps and holes in the network will appear, reducing the network lifetime.

3. ALABAMO: A Load Balancing Model for RPL

To solve the imbalance problem of RPL, we propose a load-balancing model called ALABAMO. Our model takes into account the traffic profile to avoid overloading the nodes with high-quality links. The implementation of ALABAMO is an objective function based on the Minimum Rank with Hysteresis Objective Function (MHROF). In our solution, every node broadcast the number of transmitted packets during the last measurement interval, which includes the generated and forwarded packets; such information is embedded into DIO messages. To choose the preferred parent, the nodes consider standard path quality metrics, such as ETX, and the packets number that each of its candidate parents has already sent.

ALABAMO uses a hysteresis mechanism similar to the one employed in MHROF to prevent unstable changes during fast fluctuations. We define two auxiliary constants to provide flexibility when weighting the parameters in our objective function.

The first constant is $MaxETX_{Ratio}$, which adjusts how parameter ETX will be used in the path metric. The ETX metric is still an important metric in our optimization, but we impose a range of ETX values in which all parents can are equally eligible to become the preferred one. During the optimization process, if the ratio of the ETX path metric of two parents is greater than $MaxETX_{Ratio}$, a node must select the parent with fewer sent packets. Considering node n and p_1 and p_2 as candidates parents of n, if ETX_{p_1} is the ETX metric of p_1 and ETX_{p_2} is the ETX metric of p_2, then the ETX metric ratio is calculated as follows:

$$ETX_{Ratio} = \begin{cases} \frac{ETX_{p_2}}{ETX_{p_1}} * 100 & \text{if } ETX_{p_1} \geq ETX_{p_2} \\ \\ \frac{ETX_{p_1}}{ETX_{p_2}} * 100 & otherwise \end{cases} \quad (1)$$

The second constant is $MaxWorkload_{Ratio}$, and it adjusts how the difference between the number of transmitted packets by different parents will affect the optimization function. We calculate the $Workload_{ratio}$ similarly to ETX_{ratio}, except that an initial offset is inserted to prevent instability at the beginning of the network execution. A common $offset$ value used in our experiments is 100. Considering $SentPkt_{p_n}$ as the number of packets transmitted by parent p_n, we have the following:

$$Workload_{Ratio} = \begin{cases} (\frac{SentPkt_{p_2}+offset}{SentPkt_{p_1}+offset}) * 100, & \text{if } SentPkt_{p_1} \geq SentPkt_{p_2} \\ \\ (\frac{SentPkt_{p_1}+offset}{SentPkt_{p_2}+offset}) * 100, & otherwise \end{cases} \quad (2)$$

The main piece of code implemented in ALABAMO is the function responsible for choosing the preferred parent, whose pseudocode is shown in **Algorithm 1**. **Algorithm 1** is executed for each pair of neighbors, in order to rank all possible parents.

Algorithm 1 is executed to analyze the parents pairwise. It receives as argument both parents and the pre-calculated workload ratio and the ETX ratio. Normally, the RPL would choose the parent with small path metric (only accounting for ETX); However, ALABAMO's mechanism checks the proportion of transmitted packets and the ETX metric. If the ETX metric of some parents are within an acceptable range and the difference of sent packets is substantial (larger than a certain threshold), the node chooses the parent with smallest number of transmitted packets.

As a result, the nodes in the network select the parent with a tolerable link quality and with less packets transmitted, increasing the load balancing. In our proposal, there is a trade-off between load balancing and packet loss. By tuning the constants $MaxETX_{Ratio}$ and $MaxWorkload_{Ratio}$ we can prioritize each of these two factors. If we reduce $MaxETX_{Ratio}$ the network will be more balanced, but packet loss will increase because more nodes with low-quality links will be selected as preferred parents. Besides, reducing the $MaxETX_{Ratio}$ increases the number of parent switches, which results in a larger number of packets transmitted to update the DODAG, consuming more energy.

$MaxWorkload_{Ratio}$ plays a secondary role and is important to prevent churning and unnecessary network overloading. If there is more than one candidate parent that could be equally selected according to the ETX metric, we select the one with lowest workload, as long as the cost of switching is not high. If we increase $MaxWorkload_{Ratio}$ we have more freedom to switch. In other words, we will select a better parent even if it is not so much better than the current one. If $MaxWorkload_{Ratio}$ is reduced it becomes harder to switch between parents, but it may be beneficial if the network is very dynamic and thus making too many switches, incurring a large cost.

Algorithm 1 Best parent selection algorithm

Require: $p_1, p_2, Workload_{ratio}, ETX_{ratio}$
Ensure: $best\_parent$

1: $min_{diff} \leftarrow rpl\_dag\_mc\_etx\_divisor \div parent\_switch\_threshold\_div$
2: $m_{p_1} \leftarrow calculate\_path\_metric(p_1)$
3: $m_{p_2} \leftarrow calculate\_path\_metric(p_2)$
4: **if** $(p_1\ OR\ p_2\ are\ current\_preferred\_parent)$ **then**
5: **if** $(m_{p_1} < m_{p_2} + min\_diff)\ AND\ (m_{p_1} > m_{p_2} - min\_diff)$ **then**
6: **if** $(SentPkt_{p_1} > SentPkt_{p_2})\ AND\ (Workload_{ratio} < MaxWorkload_{Ratio})$ **then**
7: $return\ p_2$
8: **end if**
9: **else**
10: **if** $(SentPkt_{p_2} > SentPkt_{p_1})\ AND\ (Workload_{ratio} < MaxWorkload_{Ratio})$ **then**
11: $return\ p_1$
12: **end if**
13: $return\ current\_preferred\_parent$
14: **end if**
15: **end if**
16: **if** $m_{p_1} < m_{p_2}$ **then**
17: **if** $(Workload_{ratio} < MaxWorkload_{Ratio})\ AND\ (ETX_{ratio} > MaxETX_{Ratio})$ **then**
18: $return\ p_2$
19: **end if**
20: $return\ p_1$
21: **else**
22: **if** $(Workload_{ratio} < MaxWorkload_{Ratio})\ AND\ (ETX_{ratio} > MaxETX_{Ratio})$ **then**
23: $return\ p_1$
24: **end if**
25: $return\ p_2$
26: **end if**

Therefore, the optimal value of the ETX_{ratio} and $MaxWorkload_{Ratio}$ should be set according to the network scenario. Some factors that have influence in these parameters choice are: link quality variance, packet transmission rate and network density. In our experiments we fixed $MaxWorkload_{Ratio}$ to 70 and varied ETX_{ratio} between 50 and 90 to evaluate its impact.

At line 1 of **Algorithm 1**, a hysteresis constant is set based on two other parameters. These parameters are related to how the ETX metric is calculated and its range. In our experiments the value was 128. The ETX metric of p_1 and p_2 are assigned at lines 2 and 3. Lines 4 through 15 are responsible for the hysteresis mechanism. If the difference of the two calculated metric is less than the min_{diff}, the function returns the current preferred parent and the network does not change. However, if the current preferred parent has sent more packets than the other one, the function returns the parent with less transmitted packets. It represents the load balancing mechanism. This can be seen at lines 6, 7 and 9-11.

At line 16, the algorithm checks if the metric of p_1 is smaller than p_2. Line 17 implements the balanced mechanism. Even the metric of p_1 is smaller than metric of p_2, the algorithm returns p_2 if the $Workload_{Ratio}$ is smaller than $MaxWorkload_{Ratio}$ and ETX_{ratio} is greater than $MaxETX_{Ratio}$. That means that the ETX metric of p_1 and p_2 are close and the number of transmitted packets by each is significantly different. Lines 21 through 26 work similarly to lines 17 through 20 but in accordance with the other case; when the metric of p_2 is smaller than the metric of p_1.

4. Performance Evaluation

We evaluated the ALABAMO performance and compare it to the default MHROF [Gnawali and Levis 2012], which is implemented in ContikiRPL [Tsiftes et al. 2010]. Both ALABAMO and MHROF were used in a real network with nodes running ContikiOS.

In a wireless sensor network, the nodes that are first to deplete their battery are the ones closer to the root, assuming that all nodes have the same transmitting power and battery size, and that the network traffic is uniform. Hence, it is important that all sub-trees have an evenly distributed number of nodes.

4.1. Evaluation scenario and performance metrics

The scenario used for the real experiments consisted of 41 nodes in the Tutornet testbed; one sink and 40 sensor nodes. The motes are deployed in an area of 55 x 30 meters, and placed above the ceiling. The testbed is located in a working-environment with 7 laboratories and more than 10 working offices, all of them separated by dry-walls. There are 8 Wi-Fi access points spread on the floor, using the whole 2.4 GHz spectrum, and various other sources of interference, such as Bluetooth devices and micro-waves. Moreover, it can be estimated that more than 60 people work or pass by the considered floor daily, which creates a real dynamic and challenging environment for wireless networking. The location of the nodes can be found at the Testbed's website at *http://anrg.usc.edu/www/tutornet/*. In the experiments the sink node was *Node 1* and the 40 sensor nodes were *Nodes 2 to 41*, in the topology shown on the mentioned website. All nodes are TelosB motes running Con-

tikiOS 3.0. ALABAMO was implemented using the interfaces provided by ContikiOS [1].

Each sensor node transmits one UDP packet towards the sink every 30 seconds. Each experiment lasts for 2 hours, summing up to $9,600$ data packets per experiment. In addition to data packets there are also regular ICMPv6 packets used by RPL protocol. In all experiments DIO and DAO maximum timeout was set to $1,048$ seconds. The utilized Medium Access Control (MAC) layer has a maximum of 3 link-layer retrials.

In the experiments, we disregarded issues with nodes synchronization in the MAC layer and assumed that nodes are capable of sleeping in order to save energy. Hence, energy consumption is restricted to packets sent, received and retransmitted. We found out throughout our experiments that X-MAC implementation in ContikiOS does not guarantee highly precise synchronization using TelosB motes, so we decided to employ NullMAC, with no sleep functionality, and calculate the total energy consumption using the packets statistics. It is important to emphasize that this synchronization problem has already been solved by standard IEEE 802.15.4e and its protocols DSME and TSCH. Operating systems such as OpenWSN [2] that are focused on critical applications provide MAC layers that guarantee a minimum waste of energy due to desynchronization and overhearing.

The experiments considered the default **MHROF** as found in ContikiOS 3.0 implementation. ALABAMO proposal with $MaxWorkload_{Ratio}$ equals to 70 when the constant $MaxETX_{Ratio}$ is set to either 80 or 90. Hence, we used names **ALABAMO-80** and **ALABAMO-90** for differentiating the proposal with two different constants.

Each experiment was repeated 12 times (each lasting 2 hours) in order to calculate standard deviation and confidence intervals.

4.2. Routing tree

We first analyzed the impact of ALABAMO on the routing tree. Since it tries to balance out the number of packets forwarded by intermediate nodes, it tends to result in sub-trees with approximately the same number of nodes and a shallow routing tree (with smaller maximum hop count), since nodes have more flexibility to use links with lower ETX to reach the sink.

In the experiments, a snapshot of the routing tree was taken every 30 minutes. Table 1 shows the average size of all sub-trees and the average size of the heaviest sub-tree. The heaviest sub-tree is the one with the most nodes connected to it. It is of special interest because it is usually the first one to disrupt due to battery depletion of the head node.

Model	Sub-trees size	Std deviation	Heaviest sub-tree size	Std deviation
MHROF	4.00	4.93	18.00	0.12
ALABAMO-80	3.63	3.79	11.93	1.67
ALABAMO-90	3.64	3.34	10.33	1.23

Table 1. Average size of sub-trees and average size of heaviest sub-tree.

[1]Code can be found at http://anrg.usc.edu/downloads

[2]http://www.openwsn.org

The experiments showed that MHROF has the most loaded sub-trees, with 4 nodes on average. Both of ALABAMO solutions have a similar average of sub-trees with 3.6 nodes, which is less than the MHROF. An important aspect to be considered is the standard deviation; both ALABAMO solutions decreased the standard deviation of sub-tree sizes (when compared to MHROF), which means that the distribution of nodes in the sub-trees is more uniform, favoring a better network load balancing.

Concerning the heaviest sub-trees, we clearly notice the critical imbalance problem in MHROF, which has sub-trees with 18 nodes on average, corresponding to almost half of the nodes in the network. ALABAMO reduced the heaviest sub-tree size by at least 6 nodes in comparison to MHROF.

We also calculated the maximum hop-count of the network, which is the number of hops from the furthermost node from the sink. MHROF and ALABAMO-90 had both the same maximum hop count of 4 in all experiments. In ALABAMO-80, the maximum hop count fluctuated between 3 and 4, with an average of 3.5 and standard deviation of 0.52. Lower maximum hop count results in less packets being forwarded, which may increase the lifetime of the network and the overall delivery ratio. The solution ALABAMO-80 showed results better than MHROF's, which highlights the benefits of the proposed scheme.

4.3. Network Delivery Ratio

The delivery ratio of a network is the percentage of packets sent by the sensor nodes that are actually received by the sink. ALABAMO presents a trade-off between ETX and the parent's workload; it picks links with lower quality for better balancing, but compromises the network delivery ratio.

All packets transmitted were UDP datagrams, hence no transport-layer retransmission was employed. This way, we intend to verify the impact of using links with lower quality in the overall performance; transport-layer retransmissions would influence the delivery ratio and make it more difficult to extract such statistics. The link layer used acknowledgement packets and a maximum of 3 retransmissions. It is important to notice that, because of the light traffic, there is no loss due to queue dropping. In fact, during the experiments, we monitored the number of packets dropped and confirmed that only in a few round there was at most 1 packet dropped by some of the nodes.

Figure 1 shows the average network delivery ratio. As expected, the ALABAMO's delivery ratio is lower than MHROF's since the nodes can select parent with lower quality links. The difference between MHROF and ALABAMO-80 is approximately 22%, which is acceptable in certain applications and does not incur very large overhead if transport-layer retransmissions are employed. Since ALABAMO-90 is expected to choose links with higher ETX than ALABAMO-80, the delivery ratio of the former was supposed to be higher. However, ALABAMO-90 resulted in trees with more hops, which increases the number of packets being forwarded, and consequently, the chances of transmission failures is also increased. Indeed, results show that ALABAMO-90's delivery ratio is lower than ALABAMO-80's. This fact stresses that using links with lower ETX may result in improvements in various aspects of the network, with regard to reducing energy consumption and even creating a more efficient routing tree. The obtained delivery ratio on all three scenarios is sufficient for most stateless Restful applications based on protocols

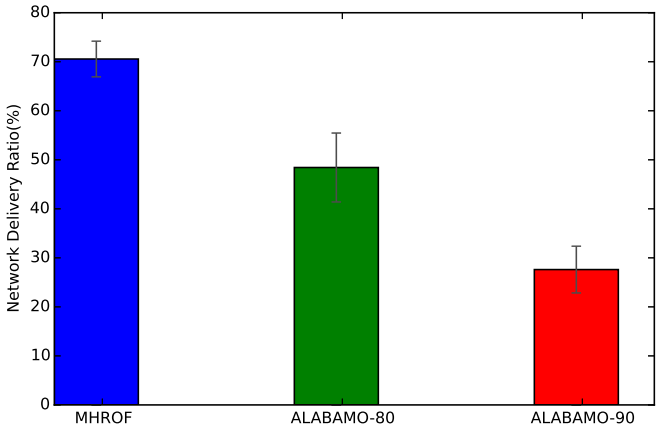


Figure 1. Network delivery ratio (percentage of data packets received by the sink).

such as CoAP, where retransmissions of missed requests are expected to happen.

4.4. Energy consumption and network lifetime

The energy consumption of a node can be split into processing (microcontroller and sensor) and communication. In general the communication consumption is the highest burden for achieving a longer lifetime. Energy is consumed by the radio chip when it is transmitting, receiving and hearing the medium. We disregard the waste due to overhearing. Hence, our calculations consider energy consumed due to reception of messages to be forwarded (since all messages are destined towards the sink) and transmission of generated and forwarded messages.

We measured the power consumed during two hours of experiment. The lower the standard deviation, the more balanced is the network. In addition, the more balanced is the network, the longer is its lifetime. Table 2 shows the average power consumption of all nodes and the average power consumption of the most loaded node (the head of the heaviest sub-tree).

Figure 2 shows that expected lifetime of the network. For this estimation, we assume that the most loaded node is the first node to deplete its battery, causing the network disruption. The bar graphs shows the lifetime normalized by the shortest one (MHROF).

Model	Average consumption (per node in mW)	Std. deviation	Average consumption (most loaded node in mW)	Std. deviation
MHROF	4.65	6.24	34.72	9.04
ALABAMO-80	4.07	3.08	16.69	3.70
ALABAMO-90	4.04	4.79	21.21	15.12

Table 2. Average power consumption of all nodes and of the most loaded node.

We noticed that all models have similar average consumption, which is expected, since the same number of packets are generated in the experiments. However, we highlight the difference of the standard deviation values. ALABAMO-80 has the lowest value. MHROF builds an unbalanced network and its standard deviation value is the highest. In

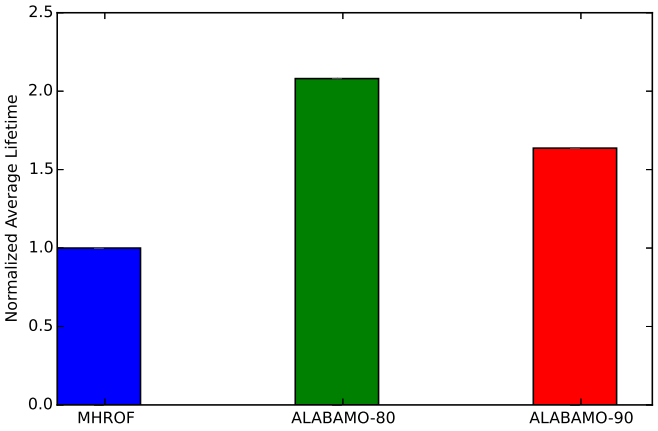


Figure 2. Normalized average network lifetime.

regards to the network lifetime, we can verify that ALABAM0-80 has a two-fold increase related to MHROF, while ALABAMO-90 has a network lifetime slightly lower than the ALABAMO-80.

4.5. Parent switching

The flexibility of dynamically choosing parents with lower workload comes with a cost. Every time the preferred parent of a node is changed, the routing tree is disturbed and nodes need to learn the new preferred paths.

MHROF has the smallest average number of parent switches per node, with an average of 0.30 and small standard deviation of 0.69. On the other hand, ALABAMO-90 almost doubled the number of parent switches (average of 0.59), and ALABAMO-80 increased in more than 4 times this number (average of 1.24). The standard deviation of the statistics of ALABAMO solutions was also much higher, more than 1.2. It is clear that even though MHROF cannot form balanced trees, it generates trees that do not change over time, which can be beneficial for critical applications that cannot afford network churn.

5. Related Work

Load-balancing in LLNs has been widely investigated over the last few years. Some hierarchical solutions achieve a balanced network, which organize the nodes into clusters and selecting the best arrangement and cluster heads [Kumar et al. 2011]. On the other hand, flat-based routing solutions are more suitable for non-uniform deployment and heterogeneous traffic pattern. Workload balancing with flat routing algorithms is achieved through the construction of balanced routing trees. Such trees may be constructed either centrally [Durmaz Incel et al. 2012] or in a distributed fashion [Bhatti et al. 2013, Kim 2015]. The former type requires global knowledge of the network and is less flexible for mobile or heterogeneous networks, but usually obtains best results; the latter is based on local information and is more adaptable to network changes but, in general, presents non-optimal results.

Bhatti et al. (2013) proposed a solution where nodes with heavy workload signal their status by delaying the transmission of DIO messages. Based on the past DIOs

time-stamps, each node can dynamically decide how to spread out data among the candidate parents. Although the implicit signaling does not require any information to be transmitted, it might be easily affected by fading links that can increase packet loss and then be interpreted as an overloading signal. The proposal was only evaluated on the NS2 simulator.

Gaddour et al. (2014) proposed a holistic OF that combines four metrics: hop count, link quality, node energy and end-to-end delay, using fuzzy parameters. Although this type of composite solution is flexible for different scenarios, it is not clear how applications would signal to the underlying routing layer which metric should be optimized. Beside this issue, RPL protocol defines that there may exist multiple DODAGs in a network, each one with a different OF that targets a specific application. Hence, composite metrics that consider too many metrics may incur greater complexity than the use of multiple simpler OFs. The solution in [Gaddour et al. 2014] was only evaluated with the Cooja simulator [Dunkels et al. 2004].

Some recent works [Capone et al. 2014, Iova et al. 2014, Kim 2015] have designed OFs that optimize the network lifetime considering two different metrics, one aimed at choosing links with good quality and the other for saving energy. [Capone et al. 2014] presented two different ways of estimating the residual energy of nodes and use this model to calculate a lifetime cost metric. The residual energy concept was also used by Iova et al. (2014), but rather than equally saving energy among all nodes, their proposed metric focused on energy consumption of the bottleneck nodes. It also considered parameters such as link data rate, different power transmission, and link qualities. Both solutions [Capone et al. 2014, Iova et al. 2014] present good results but were exclusively evaluated using simulation models. Even these are distributed solutions, they rely on models and equations that require parameters that may be different for each node, such as energy source (battery size) and transmission power.

ALABAMO works primarily based on hardware-independent information, such as the number of transmitted and forwarded packets. It is important to highlight that, although many proposals use residual energy as metric, it is not well stated how they can be implemented in a distributed scenario using RPL as routing protocol. Plus, our evaluation process is based on a real network in a dynamic environment.

A proposal that is closest to ALABAMO is that of Kim (2015). His approach, named QU-RPL, shows that under heavy traffic LLNs suffer from a serious load balancing problem that cause network lifetime to be drastically reduced. QU-RPL, like ALABAMO, takes into account the number of packets being forwarded by the candidate parents. However, while QU-RPL optimizes the packet delivery ratio and is best employed in cases where queue utilization is high, ALABAMO aims at extending the network lifetime and may be used in both heavy and light traffic conditions.

6. Conclusion

Here we present a model, called ALABAMO, that implements a new Objective Function for RPL. The proposal aims at balancing the traffic among forwarder nodes and, hence, extending the network lifetime. Results show that ALABAMO can reduce in half the energy consumption of the most loaded node, which represents a 2-fold increase in network lifetime (considering that the network is disrupted when the first node dies). Besides, the

standard deviation of energy consumption of all nodes was also reduced in half, which means that all nodes have their battery lasting a similar amount of time, making the network last longer as a whole.

Therefore, we conclude that ALABAMO builds a better-balanced tree than the MHROF and improves the network lifetime, even though it reduces the network delivery ratio a little. Lastly, ALABAMO is a practical solution to imbalance problem and easy to implement.

Future work will compare ALABAMO with other load balanced algorithms that uses centralized and distributed techniques. Moreover, we will analyze the impact of ALABAMO in other metrics such as delay.

References

Accenture (2014). Driving Unconventional Growth through the Industrial Internet of Things.

Bhatti, G., Orlik, P., and Parsons, K. (2013). Load balanced routing for low power and lossy networks. *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2238–2243.

Capone, S., Brama, R., Accettura, N., Striccoli, D., and Boggia, G. (2014). An Energy Efficient and Reliable Composite Metric for RPL Organized Networks. *2014 12th IEEE International Conference on Embedded and Ubiquitous Computing*, pages 178–184.

Clausen, T., Verdiere, A., Yi, J., Herberg, U., and Igarashi, Y. (2014). Observations of rpl: Ipv6 routing protocol for low power and lossy networks. Internet-Draft draft-clausen-lln-rpl-experiences-09, IETF Secretariat. `http://www.ietf.org/internet-drafts/draft-clausen-lln-rpl-experiences-09.txt`.

Conta, A., Deering, S., and Gupta, M. (2006). Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. RFC 4443, RFC Editor. `http://www.rfc-editor.org/rfc/rfc4443.txt`.

Dunkels, A., Grönvall, B., and Voigt, T. (2004). Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE.

Durmaz Incel, O., Ghosh, A., Krishnamachari, B., and Chintalapudi, K. (2012). Fast Data Collection in Tree-Based Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 11(1):86–99.

Gaddour, O., Koubaa, A., Baccour, N., and Abid, M. (2014). OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol. *2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 365–372.

Gnawali, O. and Levis, P. (2012). The minimum rank with hysteresis objective function. RFC 6719, RFC Editor. `http://www.rfc-editor.org/rfc/rfc6719.txt`.

Iova, O., Theoleyre, F., and Noel, T. (2014). Improving the network lifetime with energy-balancing routing: Application to RPL. *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 1–8.

Kim, H.-S. (2015). QU-RPL : Queue Utilization based RPL for Load Balancing in Large Scale Industrial Applications. In *12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON 2015)*, pages 265–273, Seattle.

Kumar, V., Jain, S., Tiwari, S., et al. (2011). Energy efficient clustering algorithms in wireless sensor networks: A survey. *IJCSI International Journal of Computer Science Issues*, 8(5).

Levis, P., Clausen, T., Hui, J., Gnawali, O., and Ko, J. (2011). The trickle algorithm. RFC 6206, RFC Editor. `http://www.rfc-editor.org/rfc/rfc6206.txt`.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002). Wireless Sensor Networks for Habitat Monitoring. *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97.

Tsiftes, N., Eriksson, J., and Dunkels, A. (2010). Low-power wireless IPv6 routing with ContikiRPL. *Proc. 9th ACM/IEEE International Conference on Information Processing in Sensor Networks - IPSN '10*, pages 406–407.

Vasseur, J., Kim, M., Pister, K., Dejean, N., and Barthel, D. (2011). Routing metrics used for path calculation in low power and lossy networks. Internet-Draft draft-ietf-roll-routing-metrics-19, IETF Secretariat. `http://www.ietf.org/internet-drafts/draft-ietf-roll-routing-metrics-19.txt`.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). Rpl: Ipv6 routing protocol for low-power and lossy networks. RFC 6550, RFC Editor. `http://www.rfc-editor.org/rfc/rfc6550.txt`.

Trilha Principal do SBRC 2016
Sessão Técnica 16
Computação na Nuvem

Escalonamento de processos sensível à localidade de dados em sistemas de arquivos distribuídos

Bruno Hott, Rodrigo Rocha, Dorgival Guedes

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brazil

{brhott,rcor,dorgival}@dcc.ufmg.br

Resumo. *O crescimento acelerado dos dados disponíveis para pesquisas levou recentemente à popularidade de sistemas de processamento como Hadoop, Spark e outros, que se valem de sistemas de arquivos distribuídos para armazenar e acessar grandes volumes de dados. Para garantir um melhor desempenho, é importante que o processamento ocorra próximo aos dados, mas soluções atuais buscam localidade apenas no nível da rede local, não da mesma máquina. Este trabalho avalia o ganho de se colocar o processamento exatamente na máquina que contém os dados. Para isso, propomos um escalonador de tarefas sensível à localidade dos dados, integrado ao sistema de arquivos HDFS e ao escalonador YARN. Um protótipo foi implementado usando o ambiente Watershed e seu resultado comparado com outras políticas. Os resultados mostram que o escalonamento sensível ao posicionamento por máquina pode melhorar significativamente o desempenho de aplicações que utilizam o HDFS e mesmo aquelas que já se valem de uma solução que tenta garantir o acesso a dados já em memória, como o Tachyon.*

Abstract. *The accelerated growth of the volume of data available for researchers led recently to the increased popularity of distributed data processing systems like Hadoop, Spark and others. Those systems make use of distributed file systems to store, access and process big volumes of data. To guaratee better performance, it is important that the processing takes place close to the data. However, current solutions seek local network locality, not same machine locality. This work evaluates the gains of placing processing exactly in the same machine were data are. To achieve that we propose a data locality aware task placement scheduler integrated to the YARN scheduler and the HDFS file system. A prototype was implemented using the Watershed programming environment and its result is compared to other policies. Results show that scheduling based on machine-data placement can significantly improve performance of applications that use HDFS and even those that already use a memory-based data access solution, such as Tachyon.*

1. Introdução

O aumento da conectividade e da banda na Internet, combinados com a redução do custo de equipamentos eletrônicos em geral, têm causado uma explosão do volume de dados que trafegam pela rede. Ao mesmo tempo, recursos para armazenar esses dados vêm crescendo, o que levou ao surgimento de sistemas especialmente desenvolvidos para processá-los, tendo como um exemplo inicial o modelo MapReduce da Google [Dean and

Ghemawat 2008], que foi seguido por diversas implementações de código aberto, como Hadoop [White 2009], e novos modelos, como Spark [Zaharia et al. 2010b]. Além disso, tornou-se necessário uma solução para o armazenamento desse enorme conjunto de dados e sistemas de arquivos distribuídos, como HDFS [Shvachko et al. 2010] e Tachyon [Li et al. 2014], foram surgindo. Como os dados agora representam um volume muito grande e estão distribuídos por diversas máquinas em um cluster, surge o problema de levar as aplicações para perto das bases de dados de forma eficaz. Caso isso não seja feito, o preço de mover os dados pelo sistema pode ser muito alto e prejudicar o desempenho final da aplicação.

Apesar desse problema ser reconhecido, ainda existe pouco entendimento sobre a interferência da localidade dos dados no desempenho desses frameworks. Dependendo da localização, o acesso aos dados pela aplicação pode ser realizado diretamente no disco da máquina local, pela memória local, via caching, ou a partir da memória de outra máquina do cluster, via rede. Os diversos compromissos em termos de capacidade de armazenamento, tempo de acesso e custo computacional envolvidos tornam não trivial uma decisão de posicionamento. Por esses motivos, muitas vezes esses fatores podem ser melhor avaliados em tempo de execução da aplicação, quando se pode encontrar um melhor aproveitamento dos recursos do ambiente ou gastos com hardware.

Durante o trabalho de reengenharia do ambiente Watershed [Ramos et al. 2011, Rocha et al. 2016], realizamos a integração da nova implementação com o ecossistema Hadoop, utilizando o gerente de recursos YARN [Vavilapalli et al. 2013] e o sistema de arquivos HDFS. Em nossa experiência anterior com o Watershed e com seu antecessor, o Anthill [Ferreira et al. 2005], observamos sempre um impacto ao se dividir um fluxo de processamento de forma que o processamento inicial de dados lidos do disco fosse executado em um nó diferente daquele de onde os dados foram lidos. Com base nessa observação, decidimos implementar um escalonador sensível à localidade de dados, garantindo que a primeira tarefa de processamento definida para qualquer dado fosse sempre executada em uma máquina onde aqueles dados estivessem disponíveis localmente, seja em disco ou em memória.

Este trabalho avalia os aspectos de localidade em ambientes de processamento de dados massivos durante o desenvolvimento do escalonador. Com base na descrição da tarefa de processamento, que indica as etapas de processamento a serem executadas e o(s) arquivo(s) de entrada a ser(em) utilizado(s), das informações providas pelo HDFS (ou Tachyon) sobre a localização dos dados no sistema e dos recursos oferecidos pelo gerente de recursos YARN, nosso escalonador, sempre que possível, executa tarefas em máquinas onde aquele dado esteja acessível, seja em memória ou disco. Realizamos diversos experimentos com o intuito de comparar os resultados com outros escalonamentos possíveis. Os resultados obtidos comprovam as vantagens de se levar em conta o posicionamento dos dados no escalonamento de aplicações desse tipo.

Com isso em mente, o restante deste artigo está organizado da seguinte forma: a seção 2 fornece mais detalhes sobre os elementos do ecossistema Hadoop utilizados (HDFS, YARN e Tachyon), com base nos quais a seção 3 descreve a implementação do nosso escalonador, cujos resultados de avaliação são apresentados na seção 4. Finalmente, a seção 5 discute trabalhos relacionados e a seção 6 conclui com algumas observações finais e discussão de trabalhos futuros.

2. Elementos do ecossistema Hadoop

No desenvolvimento do nosso escalonador, três componentes do ecossistema Hadoop são de particular importância para o entendimento da solução adotada [Murthy et al. 2013]. Na seções a seguir, apresentamos os principais conceitos relacionados ao YARN, HDFS e Tachyon.

2.1. Serviço de gerenciamento de recursos e processos distribuídos

Idealmente, aplicações distribuídas decidem dinamicamente em qual nó de computação cada tarefa será executada, monitorando o estado de cada tarefa durante a execução. Monitorar o estado das tarefas possibilita, por exemplo, saber se as tarefas foram concluídas, bem como tomar medidas de tolerância às falhas sempre que necessário. Para ser capaz de realizar um escalonamento inteligente dessas tarefas distribuídas, considerando maior nível de detalhes do ambiente de execução, é necessário também gerenciar os recursos disponíveis oferecidos pelos nós de computação.

O YARN, escalonador de recursos do ecossistema Hadoop, oferece o serviço de uma plataforma básica para a gestão das aplicações distribuídas em um nível mais alto de abstração. Permite, assim, que diferentes aplicações distribuídas possam coexistir em um mesmo ambiente de execução em *cluster*. A unidade de alocação seria um *contêiner*, uma combinação de CPU, memória e disco, assinalada para executar o processamento de uma tarefa.

A principal ideia do YARN é particionar suas duas principais responsabilidades — gerenciamento de recursos e escalonamento de tarefas – em componentes separados: um gerenciador global de recursos (*ResourceManager* - RM) e uma tarefa mestre de gerenciamento por aplicação (*ApplicationMaster* - AM). O sistema YARN é composto pelo *ResourceManager* e uma tarefa escrava em cada nó do cluster, chamada de *NodeManager* (NM) [White 2009].

O *ApplicationMaster*, que é o módulo desenvolvido especificamente por cada aplicação, é o processo que coordenada a execução da própria aplicação no ambiente do *cluster*. O *NodeManager* é responsável por gerenciar a execução dos contêineres locais, monitorando suas utilizações de recursos — tais como CPU, memória, disco e rede — enviando para o RM informações sobre os contêineres. O *ResourceManager* é a autoridade máxima que arbitra a divisão dos recursos entre todos os aplicativos em execução no ambiente. O *NodeManager* tem um componente para o escalonamento de recursos, que é responsável pela alocação de recursos para os vários aplicativos em execução sujeito às restrições principais de capacidade, prioridade, e outros fatores. O escalonador não realiza nenhum monitoramento ou rastreamento para a aplicação, sem garantias de reiniciar tarefas que não são adequadamente finalizadas devido a qualquer falha da própria aplicação ou falhas de hardware. O escalonador executa seu escalonamento baseado em recursos requisitados pelo aplicativo usando a noção abstrata de um contêiner de recursos. A figura 1 ilustra o funcionamento do YARN no disparo de aplicações, cada uma com seu *Application Master*.

O escalonador discutido neste trabalho tira proveito do YARN para obter as informações sobre recursos disponíveis (nós de processamento), mas processa a informação de forma a garantir que os nós selecionados para processamento contenham

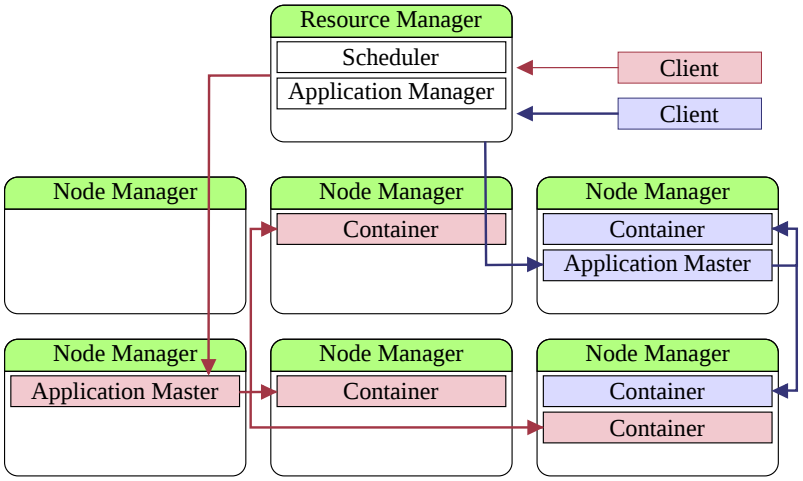


Figura 1. YARN gerenciando duas aplicações distintas em um ambiente de cluster.

os dados a serem processados. Isso é feito utilizando a informação fornecida pelos sistemas de arquivos HDFS/Tachyon (dependendo da configuração da aplicação). Ao disparar os processos da aplicação, cada processo é configurado com a informação necessária para que ele tenha acesso aos dados locais naquela máquina.

2.2. Sistema de arquivos distribuído do Hadoop

Sistemas para processamento de dados massivos lidam com coleções de dados que podem esgotar toda a capacidade de armazenamento de uma única máquina. Além disso, processar tais coleções de dados demanda grande poder computacional. Por esse motivo, grandes coleções de dados são armazenadas de maneira distribuída em diversos nós computacionais, permitindo que grandes volumes de dados sejam armazenados, além da capacidade individual de cada máquina, bem como o processamento e acesso distribuído de tais coleções de dados.

O *Hadoop Distributed File System* (HDFS) [Shvachko et al. 2010] foi baseado principalmente no *Google File System* (GFS) [Ghemawat et al. 2003], que é um sistema distribuído de arquivos que proporciona tolerância às falhas mesmo enquanto executando sobre hardware considerados como produtos de conveniência. Nesse sistema, arquivos são escritos sequencialmente quando criados, ou dados são acrescentados sequencialmente ao final de um arquivo já existente, pela operação de *append*[1].

O HDFS é responsável por armazenar metadados sobre o sistema de arquivo bem como arquivos de usuários. Arquivos de metadados são armazenados em um servidor dedicado, chamado de *NameNode*, enquanto os arquivos de usuários são armazenados em outros servidores chamados de *DataNodes*. Todos os arquivos têm seus dados divididos em blocos grandes, que são distribuídos pelos discos dos *DataNodes*. Cada bloco é também replicado em outros *DataNodes* por motivos de confiabilidade na recuperação da informação. Além de garantir durabilidade dos dados, esta estratégia tem a vantagem adicional de que a largura de banda para transferência de dados é ampliada, e há mais

[1]As primeiras versões do HDFS não permitiam a operação de *append*, porém essa funcionalidade foi adicionada em versões posteriores (versão `0.20-append` ou `0.20.2`)

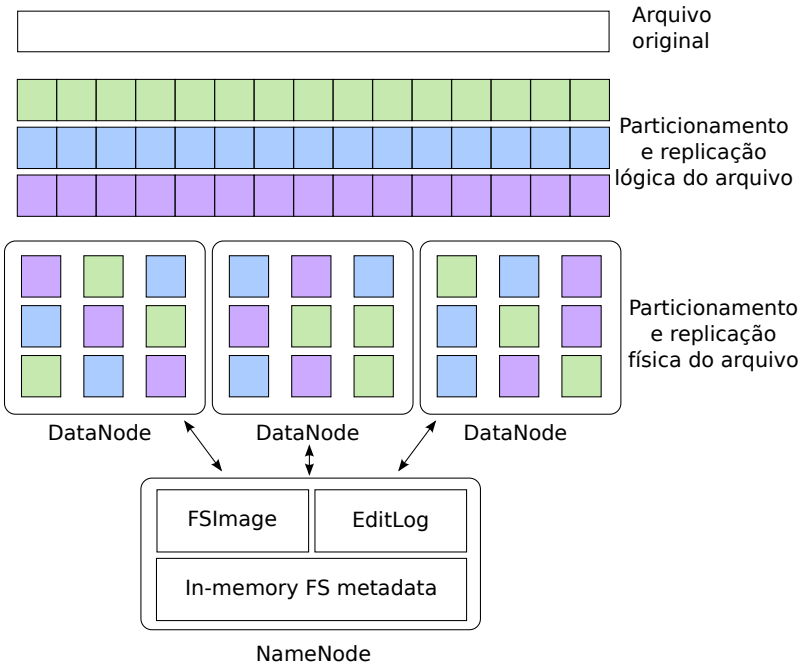


Figura 2. Visão geral da organização do HDFS, ilustrando o particionamento e replicação distribuída de um arquivo.

oportunidades para escalonar a computação para perto dos dados necessários. A figura 2 ilustra como um arquivo é particionado em blocos de tamanhos iguais (geralmente 64 ou 128 MB), de maneira que cada bloco é replicado nas diferentes máquinas que compõem o cluster.

Todos os servidores são totalmente conectados e se comunicam uns com os outros por meio de protocolos baseados em TCP. Para realizar operações de leitura ou escrita de dados, um cliente requisita ao *NameNode* quais *DataNodes* devem ser acessados e, em seguida, interage diretamente com cada *DataNode*, nunca realizando tais operações por intermédio do *NameNode*.

Quando um arquivo mantido no HDFS é identificado como parte de uma aplicação, o escalonador consulta o HDFS para obter a lista de blocos daquele arquivo. Essa lista indica, para cada bloco, quais nós possuem réplicas do mesmo. De posse dessa informação, o escalonador faz a associação de nós de processamento com os blocos do HDFS que eles devem processar.

2.3. Tachyon

Os autores do Tachyon [Li et al. 2014] argumentam que os conjuntos de dados de interesse a qualquer instante (*working set*) na maioria dos datacenters são relativamente pequenos se comparados ao conjunto total de dados neles existentes.

Frameworks existentes armazenam dados intermediários em memória com a tarefa (job) e armazenam os dados de entrada como cache em memória. Porém uma operação permanece sem executar em memória, que é o compartilhamento de dados entre as tarefas. Em particular, armazenar a saída de uma tarefa com segurança com a intenção de compartilhá-la com outras tarefas é um processo lento, já que o dado é replicado através

da rede e discos para tolerância às falhas. Isto faz com que os sistemas de armazenamento em cluster atuais sejam ordens de magnitude mais lentos do que escrever em memória. Esta é a principal motivação dos criadores do Tachyon. Nesse sentido, aquele sistema foi desenvolvido para oferecer um acesso eficiente aos dados de interesse em um certo momento, mantendo-os em memória, mas com uma forma eficiente de acessar o disco para leitura e escrita, quando necessário.

Do ponto de vista estrutural, Tachyon utiliza uma arquitetura padrão de mestre-escravo similar ao HDFS, onde cada *worker* gerencia blocos locais e os compartilha com as aplicações através da memória local.

Cada *worker* executa um daemon que gerencia os recursos locais e reporta periodicamente seu status para o master. Além disso, cada *worker* usa a RAM para armazenar arquivos mapeados em memória. Dessa forma uma aplicação com localidade de dados pode interagir à velocidade da memória.

Do ponto de vista do nosso trabalho, Tachyon pode ser visto como uma cache com características semânticas especializadas para lidar com sistemas de arquivos distribuídos. O nosso escalonador foi desenvolvido para também interfacear com esse sistema, a fim de poder explorar não só a localidade em disco, mas também aquela em memória, potencialmente agregando o benefício da localidade ao acesso em memória local, ao invés de exigir uma transferência pela rede.

3. Implementação

A execução de uma aplicação dentro do ecossistema Hadoop, seja ela baseada no Hadoop MapReduce, Spark ou outros ambientes de programação, implica na definição de pelo menos duas informações principais: a identificação dos dados de entrada a serem processados (p.ex., o arquivo de entrada) e das tarefas que devem ser executadas (inclusive quantas instâncias de cada tarefa devem existir).

De posse dessas informações, o escalonador proposto no presente trabalho realiza uma consulta ao sistema de arquivos distribuído (HDFS/Tachyon) para obter a localização dos blocos (incluindo todas as réplicas) que armazenam porções dos dados de entrada. Os nós identificados são posteriormente ordenados pela quantidade de blocos existentes em cada um, de forma a priorizar os nós do cluster com a maior quantidade de blocos.

O processo de escalonamento dos blocos é dividido em duas fases, sendo a primeira responsável pela escolha das porções do arquivo que serão lidos localmente por cada instância. Para isso, são selecionados até b/i blocos locais às máquinas onde as instâncias serão executadas, onde b é o número total de blocos a serem lidos e i é o número de instâncias, considerando inclusive que podem haver mais instâncias em uma mesma máquina. Esses conjuntos são então distribuídos entre as instâncias da aplicação.

Na segunda fase, realizamos o escalonamento dos blocos que serão lidos remotamente por cada instância. Como o HDFS não garante a distribuição uniforme dos arquivos armazenados, geralmente não é possivel que toda leitura seja realizada localmente sem que o balanceamento da aplicação seja prejudicado. Por esse motivo, o escalonador verifica se faltam blocos para serem escalonados nos nós e, caso houverem, seleciona aqueles que não foram escolhidos na primeira fase e que serão, então, lidos remotamente. Esse processo é repetido até que todas as instâncias tenham recebido todos os blocos

devidamente.

O escalonador então utiliza um mecanismo de requisição sem relaxamento na localidade dos contêineres YARN, forçando que os mesmos sejam alocados considerando exatamente as localizações especificadas, derivadas das informações obtidas das fontes de dados (HDFS/Tachyon).

O Tachyon possui um funcionamento conceitualmente equivalente ao HDFS, porém os blocos de um dado arquivo podem em qualquer momento ser encontrados em blocos no disco ou já carregados na memória do sistema. Dessa forma, o escalonador prioriza nós que contenham blocos do arquivo em memória principal. Caso esses blocos em memória não existam, então a localização dos dados em disco é considerada.

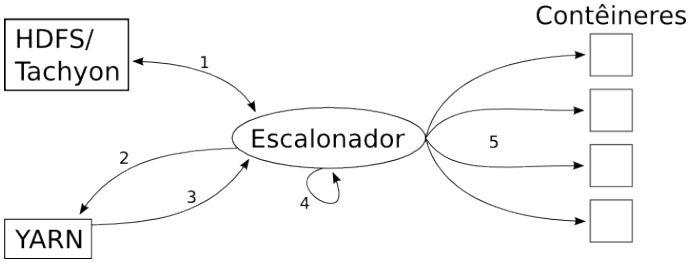


Figura 3. Passos de operação do escalonador: (1) busca do arquivo no HDFS/-Tachyon com lista de blocos, (2) pedido dos contêineres, (3) recebimento dos contêineres fora de ordem, (4) organização dos contêineres em função dos blocos e (5) execução dos processos da aplicação.

A figura 3 ilustra os passos de operação do escalonador desenvolvido. Primeiramente, o escalonador acessa o sistema de arquivo e obtém as informações sobre os arquivos envolvidos, contendo a lista de blocos e a identificação da localização de cada réplica dos mesmos. De posse dessas informações, é possível solicitar ao YARN os contêineres nos locais exatos desejados. Pela forma de operação do YARN, os contêineres para as solicitações enviadas podem ser retornados fora de ordem, então o escalonador deve processar cada resposta para identificar o nó obtido e associá-lo aos blocos presentes naquele nó. Finalmente, os processos podem ser disparados nos diversos nós com a informação sobre quais blocos eles devem acessar (que serão sempre blocos locais).

Utilizamos essa heurística de escalonamento para mantermos maior controle sobre a alocação das instâncias em relação aos seus respectivos dados de entrada e assim melhor avaliarmos o impacto da localidade de dados. Como trabalhos futuros pretendemos utilizar novas heurísticas que também levem em conta outros aspectos como a carga de trabalho atual de cada máquina.

4. Avaliação experimental

Para avaliar o desempenho do escalonador, executamos uma aplicação que lia os dados de um arquivo de entrada e computava estatísticas simples sobre o mesmo. Além do escalonador proposto, a aplicação foi executada ainda com 3 outras políticas de escalonamento: sem nenhum escalonador além do que já é provido pelo YARN, com um escalonamento aleatório e com um escalonamento que buscava representar o pior caso, onde cada processo sempre era executado em uma máquina diferente daquela que continha os dados a

serem processados. Nas figuras a seguir, os resultados do nosso escalonador são identificados pela sigla "esc", enquanto as demais situações são identificadas pelas siglas "sec", "ale" e "pes", respectivamente.

Os testes foram executados em um cluster com 10 máquinas virtuais (VMs), onde cada uma delas possuía 2 VCPUs de 2,5 GHz e 4 GB de memória RAM. Cada máquina virtual foi associada a um disco na própria máquina física em que a VM foi criada, para garantir o controle preciso de localidade física dos blocos. Todos os testes foram executados cinco vezes. Os tempos de execução apresentados se referem à média das execuções; o desvio padrão de cada conjunto de execuções é sempre apresentado nas figuras.

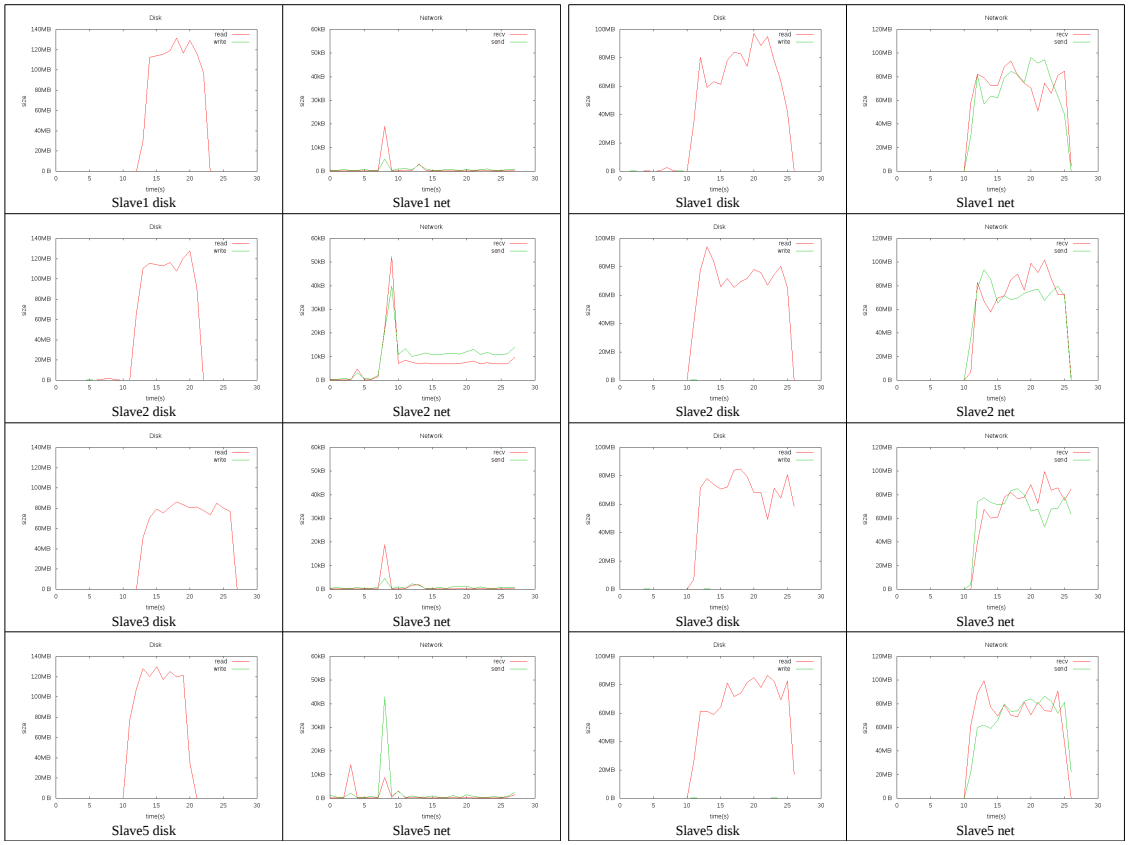
Para confirmar se o escalonador estava operando corretamente, utilizamos o comando *dstat* para coletar dados sobre o tráfego de disco e de rede em cada máquina durante os testes. Os dados coletados para essas duas grandezas comprovam que, para o escalonador proposto, os nós apresentavam um acesso a disco condizente com os dados locais a serem lidos e não havia tráfego de rede significativo, além de operações de sincronização no ambiente de processamento. Já no caso da política que sempre assinalava um processo para executar em um nó onde os dados não estavam, os dados mostravam que o padrão de acesso a discos era o mesmo do tráfego de rede enviado pelo nó — isto é, todas as leituras de disco eram destinadas a atender um processo em outro nó. Ao mesmo tempo, o padrão de dados lidos pela rede correspondia ao padrão de leitura de disco na máquina que continha os dados exigidos pelo processo da aplicação executando naquele nó. A figura 4 ilustra uma execução com quatro nós para os dois casos.

Para avaliar o comportamento do escalonamento sensível à localidade dos dados, executamos o programa de leitura de arquivos em diversas situações e medimos o tempo de execução total. Dois elementos importantes na análise foram o tamanho do cluster considerado e o tamanho do bloco usado no sistema de arquivos distribuído. O tamanho do arquivo usado era determinado em termos de número de blocos, então experimentos com tamanho de blocos maiores liam arquivos proporcionalmente maiores. Todos os arquivos foram gerados com fator de replicação 2.

Nos testes utilizando o HDFS, para que a cache do sistema de arquivos do sistema operacional não interferisse nos experimentos, efetuamos a limpeza de cache antes de cada execução. Nos testes usando o Tachyon, como seu princípio de operação é exatamente o de tentar se aproveitar ao máximo de todas as possibilidades de cacheamento dos dados, a cache era limpa apenas antes do início de cada experimento, a aplicação era executada uma primeira vez para carregar a cache do Tachyon e depois era medido o conjunto de cinco execuções.

A figura 5 mostra o efeito da variação do tamanho do bloco para dois tamanhos de cluster diferentes (4 e 8 máquinas). Tempos de execução para o cluster maior são sempre superiores, pois há mais overhead na inicialização e na sincronização entre os processos na aplicação distribuída. Claramente, o escalonador proposto apresenta bom desempenho em todos os casos.

Para o cluster de 4 máquinas, praticamente não há diferença entre os tempos do escalonador usual do YARN e do escalonador baseado em localidade, exceto para blocos de 1GB. Observando os logs da execução, um dos motivos é que, com 4 máquinas e fator de replicação 2, o próprio escalonador do YARN já balanceava os recursos de forma



(a) Execução com o escalonador proposto (b) Execução com o escalonador sem localidade

Figura 4. Padrões de acesso a disco e de dados enviados e recebidos pela rede para uma execução de uma aplicação com quatro nós de processamento, com o escalonador sensível à localidade dos dados (a) e com uma política de escalonamento que sempre posiciona o processamento em um nó diferente daquele que contém os dados (b). Pode-se ver que no primeiro caso não há tráfego de rede significativo, enquanto no segundo há sempre um volume de tráfego enviado semelhante ao padrão de leitura local.

satisfatória e tinha pelo menos 50% de chance de alocar um processador junto ao dado que ele iria processar. Já o escalonador aleatório, teve desempenho mais próximo ao pior caso por não levar em conta as informações sobre a carga dos nós que era considerada pelo YARN.

Já para o cluster de 8 máquinas, a chance de um processo ser escalonado aleatoriamente em um nó com o dado associado a ele cai para 25% e o escalonador sensível à localidade dos dados passa a ter um desempenho sensivelmente melhor, o que comprova que executar o processamento inicial dos dados na mesma máquina que os contém ainda é vantajoso em relação ao custo de acesso pela rede. Além disso, mesmo para o cluster menor, no caso de blocos de leitura/arquivos maiores, o ganho de processar os dados localmente também é visível. Na verdade, o maior ganho do escalonador foi exatamente para o cluster de 4 máquinas e blocos de 1 GB: aproximadamente 20% mais rápido que o escalonador default do YARN e duas vezes mais rápido que o pior caso. Já no cluster de 8 máquinas, o ganho de aproximadamente 20% em relação ao YARN permanece, mas os custos de sincronização com mais máquinas reduzem o ganho em relação ao pior caso.

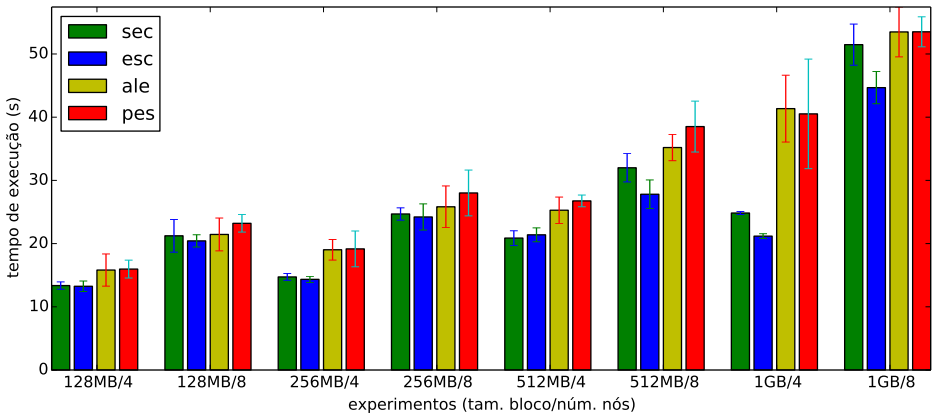


Figura 5. Resultados experimentais utilizando o escalonar integrado ao HDFS para diferentes tamanhos de blocos, diferentes números de máquinas.

Recentemente, diversos trabalhos têm sugerido a adoção de mecanismos de armazenamento em memória, como o projeto RamCloud [Ousterhout et al. 2010]. Nesse caso, o tempo de acesso e as taxas de transmissão de dados seriam aqueles da memória e não do disco, o que pode alterar significativamente o desempenho do sistema. Considerando esse cenário, o escalonador também foi configurado para trabalhar com o Tachyon, já que esse oferece o recurso de manter em memória arquivos já acessados. Como mencionado anteriormente, para os experimentos executávamos a aplicação uma primeira vez, causando a carga do arquivo para a memória do Tachyon e medíamos os tempos de cinco execuções depois que os dados já se encontravam na memória de algum nó. Devido ao comportamento do Tachyon e da aplicação, não havia replicação de blocos em memória: cada bloco era lido por apenas um processo parte da aplicação, então o dado era carregado na memória de apenas uma máquina de cada vez.

A figura 6 mostra os resultados nesse caso (não foi utilizado o tamanho de bloco de 1GB devido a limitações do Tachyon na configuração usada nos clusters do experimento). Claramente, nesse caso os ganhos do escalonador baseado em localidade são ainda mais significativos. Já que não há replicação de blocos nos dados em memória, o escalonador do YARN já não é capaz de obter o mesmo desempenho na maior parte dos casos. Além disso, o fato dos dados já estarem na memória do nó local tornam o tempo de execução bem mais regular (e menor) para o escalonador proposto. Apesar do aumento do número de máquinas do cluster ainda ter um impacto, os tempos de execução ainda caem significativamente devido às taxas de transferências mais altas.

Com o Tachyon, o escalonamento proposto é pelo menos 25% melhor que o escalonador original do YARN. Por outro lado, para o maior cluster e maior tamanho de bloco considerados, o escalonamento baseado em localidade chega a uma redução de mais de 50% em relação ao YARN isoladamente e mais de 65% em relação ao pior caso. Esses resultados indicam que, em um cenário baseado em memória, a noção de processamento dos dados na origem se torna ainda mais importante, dadas as tecnologias de rede atual.

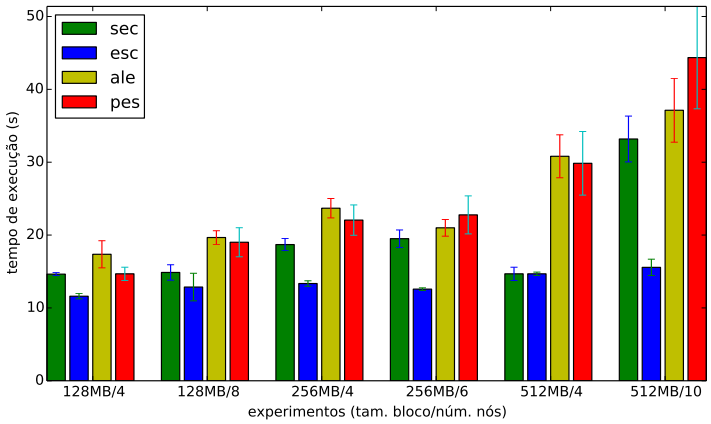


Figura 6. Resultados experimentais utilizando o escalonar integrado ao Tachyon para diferentes tamanhos de blocos, diferentes números de máquinas.

5. Trabalhos relacionados

Os artigos que descrevem Hadoop, Spark, HDFS (e GFS) e Tachyon, já mencionados, discutem alguns dos aspectos de decisão envolvidos na escolha de nós de processamento ou armazenamento em cada caso. Em geral, entretanto, o princípio de escalonamento de tarefas é o padrão do YARN, que se baseia em uma descrição simples do ambiente em termos de máquinas e racks [White 2009]. Nesse caso, processos são executados preferencialmente em uma máquina no mesmo rack onde os dados estão, mas não há um esforço maior em localizar processamento no mesmo nó que contém os dados, como neste trabalho.

Um dos primeiros trabalhos a discutir o impacto da localidade no processamento de grandes volumes de dados no contexto de datacenters modernos é devido a Barroso e Hölzle [Barroso and Hölzle 2009]. Nele os autores chamam a atenção para as diferenças em termos de acesso entre discos e memória RAM em uma mesma máquina, em um mesmo rack ou em racks diferentes. Essas diferenças, que se tornam mais significativas entre máquinas em racks diferentes, foram a base para decisões de projeto com as do Hadoop. Entretanto, os autores mostram que ainda há um degrau de desempenho significativo ao se optar por máquinas diferentes, mesmo que no mesmo rack (ligadas por um único switch de rede).

Na literatura, temos alguns escalonadores que tem como objetivo utilizar eficientemente os recursos de ambientes multiusuário. O HaSTE [Yao et al. 2014] é um escalonador integrado ao YARN, desenvolvido especificamente para aplicações MapReduce, cujo foco principal é reduzir o tempo total de conclusão de múltiplas aplicações em execução concorrente, além de também reduzir a utilização de recursos do cluster. Esse objetivo é alcançado considerando os recursos requisitados, a capacidade geral do cluster e as dependências entre as tarefas de cada aplicação. Ainda para ambientes multiusuário, Zaharia et al. [Zaharia et al. 2010a] apresentam uma técnica de escalonamento centrado em localidade de dados, chamada de *delay scheduling*, também desenvolvendo um escalonador para tarefas MapReduce no ambiente Hadoop. Os autores discutem uma heurística onde as tarefas são adicionadas à uma fila enquanto as máquinas que armazenam os da-

dos necessários estiverem ocupadas. Essa abordagem contribui para o uso equilibrado dos recursos entre os usuários ao mesmo tempo que visa localidade dos dados.

Isard et al. [Isard et al. 2009] apresentam um escalonador chamado Quincy para a plataforma distribuída Dryad [Isard et al. 2007]. O Quincy mapeia o problema de escalonamento como um grafo de fluxos, onde as arestas representam os diversos custos envolvidos no escalonamento de cada tarefa entre os diversos nós disponíveis no cluster, considerando aspectos como localidade de dados e balanceamento de carga de trabalho entre os nós.

Ousterhout et al. [Ousterhout et al. 2010] advogam o uso de sistemas de arquivos em RAM (Ramclouds) para evitar o custo do acesso a disco. Nesse sentido, aquele trabalho é ortogonal ao apresentado aqui, já que nosso objetivo é levar o processamento para a mesma máquina onde o dado se encontra, seja em memória ou disco. Os experimentos como Tachyon mostram que essa exploração do conceito de localidade pode ser até mais benéfico se o dado já se encontra em memória.

Em uma primeira análise, este trabalho se contrapõe ao trabalho de Ananthanarayanan et al. [Ananthanarayanan et al. 2011], que vai contra a preocupação com localidade de discos. Naquele artigo, os autores argumentam que diversos avanços em curso poderão vir a tornar a localidade de disco irrelevante. Nosso argumento neste artigo não tem por objetivo contrariar aquele nessa previsão. Apenas mostramos que, em condições atuais, a localidade ainda é significativa. Além disso, mostramos também que em um contexto com dados armazenados em memória, o escalonamento baseado na localidade dos dados em memória também pode resultar em ganhos de desempenho.

6. Conclusão

A popularidade do ecossistema Hadoop para o processamento de dados massivos (*big-data*) levou ao desenvolvimento de diversos sistemas de processamento baseados no sistema de arquivos HDFS e no escalonador YARN. Esses ambientes utilizam diretamente as políticas de gerência de espaço de armazenamento e escalonamento propostas pelo Hadoop, que são baseadas na noção de localidade de rack (máquinas interligadas diretamente por apenas um switch de rede).

Neste trabalho avaliamos o impacto dessa escolha comparada a uma política de alocação de recursos sensível à localidade dos dados em termos de máquinas. O argumento nesse caso é que, pelo menos para a tecnologia de Ethernet Gigabit atualmente comum na maioria dos datacenters e clusters, processar dados em uma máquina que não aquela que já os mantém gera um overhead de comunicação pela rede, mesmo que não haja gargalos entre fonte e destino. Para avaliar esse argumento, desenvolvemos um escalonador sensível à localidade dos dados integrado ao HDFS e ao YARN durante a reengenharia do ambiente de processamento Watershed.

Os resultados comprovam que o escalonamento sensível à localidade não prejudica o desempenho do sistema em clusters pequenos e com blocos menores e beneficia significativamente a execução em clusters maiores e quando os blocos de dados são maiores. Além disso, os ganhos se tornam ainda mais significativos quando os dados já estão em memória, como no caso do Tachyon. Isso pode ser um resultado ainda mais importante considerando-se propostas recentes de sistemas de armazenamento baseados em memória.

O escalonador proposto é facilmente extensível e pode ser integrado a outros ambientes de processamento que usam o ecossistema Hadoop. Como trabalhos futuros, pretendemos realizar a inclusão do mesmo no Hadoop MapReduce e no ambiente Spark, além de estender a política de escalonamento para incluir heurísticas para casos de alocação em cenários multiusuários com maior contenção de recursos.

Agradecimentos

Este trabalho foi parcialmente financiado por Fapemig, CAPES, CNPq, e pelos projetos MCT/CNPq-InWeb (573871/2008-6), FAPEMIG-PRONEX-MASWeb (APQ-01400-14), e H2020-EUB-2015 EUBra-BIGSEA (EU GA 690116, MCT/RNP/CETIC/Brazil 0650/04).

Referências

Ananthanarayanan, G., Ghodsi, A., Shenker, S., and Stoica, I. (2011). Disk-locality in datacenter computing considered irrelevant. In *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, pages 1–5, Berkeley, CA, USA. USENIX Association.

Barroso, L. A. and Hölzle, U. (2009). The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 4(1):1–108.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Ferreira, R. A., Meira, W., Guedes, D., Drummond, L. M. d. A., Coutinho, B., Teodoro, G., Tavares, T., Araujo, R., and Ferreira, G. T. (2005). Anthill: A scalable run-time environment for data mining applications. In *Computer Architecture and High Performance Computing, 2005. SBAC-PAD 2005. 17th International Symposium on*, pages 159–166. IEEE.

Ghemawat, S., Gobioff, H., and Leung, S.-T. (2003). The google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 29–43, New York, NY, USA. ACM.

Isard, M., Budiu, M., Yu, Y., Birrell, A., and Fetterly, D. (2007). Dryad: Distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2007 Eurosys Conference*, Lisbon, Portugal. Association for Computing Machinery, Inc.

Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K., and Goldberg, A. (2009). Quincy: fair scheduling for distributed computing clusters. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 261–276. ACM.

Li, H., Ghodsi, A., Zaharia, M., Shenker, S., and Stoica, I. (2014). Tachyon: Reliable, memory speed storage for cluster computing frameworks. In *Proceedings of the ACM Symposium on Cloud Computing*, SOCC '14, pages 6:1–6:15, New York, NY, USA. ACM.

Murthy, A., Vavilapalli, V. K., Eadline, D., Markham, J., and Niemiec, J. (2013). *Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2*. Pearson Education.

Ousterhout, J., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Parulkar, G., Rosenblum, M., Rumble, S. M., Stratmann, E., and Stutsman, R. (2010). The case for ramclouds: Scalable high-performance storage entirely in dram. *SIGOPS Oper. Syst. Rev.*, 43(4):92–105.

Ramos, T., Silva, R., Carvalho, A. P., Ferreira, R. A. C., and Meira, W. (2011). Watershed: A high performance distributed stream processing system. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2011 23rd International Symposium on*, pages 191–198. IEEE.

Rocha, R., Hott, B., Dias, V., Ferreira, R., Meira, W., and Guedes, D. (2016). Watershedng: an extensible distributed stream processing framework. *Concurrency and Computation: Practice and Experience*. (accepted for publication). doi:10.1002/cpe.3779.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al. (2013). Apache hadoop YARN: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM.

White, T. (2009). *Hadoop: the definitive guide: the definitive guide*. O'Reilly Media, Inc.

Yao, Y., Wang, J., Sheng, B., Lin, J., and Mi, N. (2014). Haste: Hadoop yarn scheduling based on task-dependency and resource-demand. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, pages 184–191. IEEE.

Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., and Stoica, I. (2010a). Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pages 265–278, New York, NY, USA. ACM.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010b). Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pages 10–10.

Uma Análise do Tráfego de Controle de uma Nuvem IaaS Geodistribuída

Tatiana Sciammarella[1], Rodrigo S. Couto[2], Marcelo G. Rubinstein[2]
Miguel Elias M. Campista[1] e Luís Henrique M. K. Costa[1] *

[1]Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - DEL/POLI
[2]Universidade do Estado do Rio de Janeiro - FEN/DETEL/PEL

{tatiana,miguel,luish}@gta.ufrj.br, {rodrigo.couto,rubi}@uerj.br

Abstract. *Geodistributed clouds are composed of several virtual machine servers scattered over sites interconnected by Wide Area Networks. In this scenario, the traffic generated by control messages can be prohibitive, because of the high bandwidth costs of these networks. This paper evaluates the potential of this type of traffic as an obstacle to increase the number of servers in the cloud. From the results obtained with the OpenStack orchestrator, we estimate that 100 servers and 15 virtual machines per server generates an average traffic about 2.7 Mb/s towards the infrastructure controller. To this average, we can add traffic bursts produced upon creation and destruction of virtual machines, further aggravating the problem. These results point out that, if the IaaS cloud is not well designed, the control traffic can become a concern for smaller clouds, such as collaborative and university ones.*

Resumo. *As nuvens geodistribuídas são compostas por diversos servidores de máquinas virtuais espalhados em sítios interconectados por redes de longa distância. Nesse cenário, o tráfego gerado por mensagens de controle pode ser proibitivo, visto que o custo de banda passante nessas redes é elevado. Este trabalho avalia o potencial desse tipo de tráfego como um obstáculo ao aumento do número de servidores da nuvem. A partir dos resultados obtidos com o orquestrador OpenStack, estima-se que com 100 servidores e 15 máquinas virtuais por servidor gera-se um tráfego médio de cerca de 2,7 Mb/s para o controlador da infraestrutura. A essa média pode-se somar os picos de tráfego gerados durante a criação e a destruição de máquinas virtuais, agravando o problema. Esses resultados mostram que, caso a nuvem IaaS não seja bem planejada, o tráfego de controle pode se tornar uma preocupação para nuvens de menor porte, como as colaborativas e universitárias.*

1. Introdução

A computação em nuvem vem conquistando popularidade nos mais diversos ramos produtivos por permitir a terceirização dos serviços de tecnologia da informação. Diferentes modelos de serviço podem ser oferecidos, como é o caso do IaaS (*Infrastructure-as-a-Service* - Infraestrutura como Serviço) [Jennings e Stadler, 2015], reduzindo os gastos com manutenção e administração de recursos computacionais. No modelo IaaS, o

*Este trabalho foi realizado com recursos da RNP, CNPq, FAPERJ e CAPES.

provedor de infraestrutura disponibiliza, sob demanda, máquinas virtuais (*Virtual Machines* - VMs) aos seus clientes para uso personalizado. Tais VMs podem ser acessadas facilmente pela Internet através de um navegador *web*, por exemplo. Algumas nuvens IaaS são geodistribuídas, ou seja, possuem uma infraestrutura dividida em sítios interconectados por redes de longa distância (*Wide Area Network* - WAN), o que tem reflexos em termos de resiliência e desempenho [Develder et al., 2012].

As grandes nuvens geodistribuídas são compostas por sítios autônomos interconectados, com centenas ou milhares de servidores cada um [Develder et al., 2012]. Esses servidores, cujo papel é hospedar as VMs dos clientes, são gerenciados por um orquestrador de nuvem, como o OpenStack [OpenStack, 2015] e o CloudStack [Apache, 2015]. O orquestrador executa diversas tarefas de controle da infraestrutura, como a escolha dos servidores hospedeiros de um determinado conjunto de VMs, a criação e a destruição de VMs, a autenticação de usuários e a coleta de estatísticas de uso. Geralmente, cada sítio de uma nuvem geodistribuída possui uma ou mais máquinas que atuam como controladores, executando os módulos principais de orquestradores como o OpenStack ou o CloudStack. Eventualmente, para levar em conta aspectos da arquitetura geodistribuída, os módulos do controlador de um sítio trocam mensagens com os outros sítios da nuvem.

Um cenário de utilização importante de uma nuvem IaaS geodistribuída é o compartilhamento de recursos computacionais entre diferentes entidades. Por exemplo, universidades ou centros de pesquisa possuem recursos computacionais que podem ser compartilhados através de uma única nuvem IaaS [Couto et al., 2015]. Esse tipo de serviço é denominado nuvem IaaS colaborativa, na qual diferentes entidades com interesses comuns compartilham seus recursos computacionais [Endo et al., 2011]. Uma característica das nuvens IaaS colaborativas, diferente de outras nuvens geodistribuídas, é que o número de servidores de VMs por sítio pode ser pequeno, enquanto o número de sítios pode ser muito grande. O cenário é de múltiplas instituições com recursos mais modestos em vez de uma grande corporação com centros de dados distribuídos. Sendo assim, não é desejável a instalação de um controlador para cada sítio, como ocorre em grandes nuvens geodistribuídas. Dessa forma, um único controlador centralizado pode ser utilizado para controlar os recursos de todos os sítios através de redes WAN [Couto et al., 2015]. Esse tipo de comunicação centralizada introduz por outro lado desafios na implementação de uma infraestrutura de nuvem colaborativa, principalmente em relação à escalabilidade do sistema. O tráfego de controle gerado se concentra próximo ao controlador, podendo sobrecarregá-lo. Além disso, essa maior carga de controle pode elevar os custos com banda passante, que são mais altos em redes WAN.

Assim, este trabalho analisa o impacto da comunicação entre o nó controlador da nuvem e os nós servidores de VMs na realização de tarefas de controle em uma nuvem OpenStack. O objetivo é verificar até que ponto o tráfego de controle representa um obstáculo ao crescimento de uma nuvem colaborativa. As contribuições deste trabalho são a identificação dos elementos que possam representar um gargalo na infraestrutura geodistribuída e a análise das trocas de mensagens entre o nó controlador e os servidores de VMs. A partir dos resultados obtidos é possível concluir, por exemplo, que cada servidor de VMs adicionado à infraestrutura pode gerar aproximadamente um aumento médio de 15 kb/s no tráfego da rede, enquanto que cada VM em repouso contribui com 0,77 kb/s. Consequentemente, estima-se que em uma nuvem com 100 servidores e 15 máquinas vir-

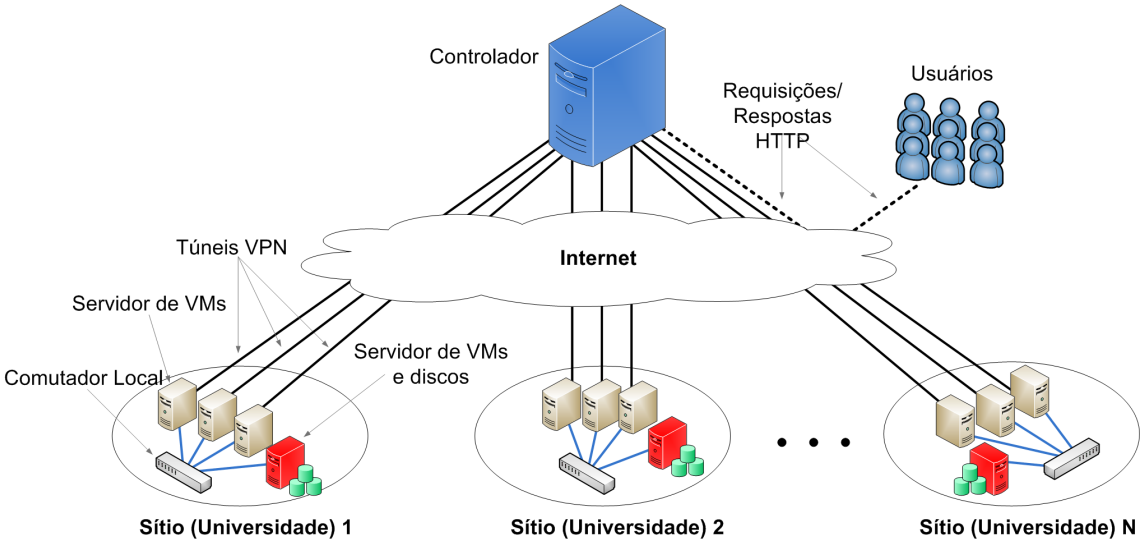


Figura 1. Arquitetura geodistribuída da nuvem colaborativa utilizada neste trabalho.

tuais por servidor, um tráfego médio de cerca de 2,7 Mb/s seria gerado para o controlador da infraestrutura. Apesar de esse valor ser pequeno para redes WANs comumente instaladas em grandes provedores de nuvem, esse tráfego pode rapidamente se tornar uma preocupação para nuvens menores, como as colaborativas e universitárias.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta a arquitetura da nuvem colaborativa. A Seção 3 descreve os detalhes relevantes do orquestrador de nuvens adotado, o OpenStack. A Seção 4 descreve a plataforma experimental utilizada para a avaliação de desempenho e analisa os resultados obtidos. A Seção 5 descreve os trabalhos relacionados, destacando a originalidade da análise realizada neste trabalho. Finalmente, a Seção 6 conclui o trabalho e indica possíveis trabalhos futuros.

2. Arquitetura da Nuvem Colaborativa

Este trabalho utiliza uma arquitetura em nuvem geodistribuída entre universidades e centros de pesquisa, proposta em [Couto et al., 2015]. O objetivo fundamental é compartilhar recursos computacionais entre as instituições envolvidas. Assim, em momentos de pico de utilização da infraestrutura local de uma instituição, seus usuários podem utilizar recursos computacionais ociosos de outras instituições, compartilhados através da nuvem. Por um lado, as instituições aumentam o seu poder computacional de maneira intrínseca ao modelo em nuvem; por outro lado, devem oferecer parte dos seus próprios recursos para que a nuvem funcione. Essa é a ideia da colaboração, calcada na possibilidade de uso intenso dos próprios recursos por parte das instituições de maneira assíncrona.

A arquitetura de nuvem colaborativa, apresentada na Figura 1, consiste em sítios, instalados em universidades ou centros de pesquisa, que são conectados a um Controlador centralizado. Cada sítio possui máquinas dos tipos Servidor de VMs e Discos e Servidor de VMs, conectadas entre si por um comutador local. Ambos os tipos de máquinas são utilizados para hospedar VMs, utilizando o KVM [Kivity et al., 2007] como hipervisor. O Servidor de VMs e Discos, por sua vez, possui a funcionalidade adicional de centralizar o disco virtual de todas as VMs de um sítio, permitindo assim a migração ao vivo de

VMs entre as máquinas do mesmo sítio [Couto et al., 2015]. O Controlador, além de realizar todas as funções de gerenciamento das máquinas dos sítios, é responsável por receber e atender requisições de criação de VMs pelos usuários. Além disso, fornece interfaces de acesso a essas VMs para os usuários. Tanto os Servidores de VMs como os Servidores de VMs e Discos estão conectados ao Controlador através de túneis VPN (*Virtual Private Network*) para atravessar a Internet. Para o gerenciamento da arquitetura da Figura 1, tanto o nó Controlador quanto as máquinas dos sítios executam o orquestrador OpenStack [OpenStack, 2015], detalhado na próxima seção.

O cenário previsto suporta a existência de sítios pequenos (p.ex., laboratórios), com disponibilidade possivelmente de apenas um servidor. Nesse caso, não é satisfatório executar uma solução completa de nuvem em cada sítio da infraestrutura, justificando a existência de um único controlador. Além disso, muitas vezes o parceiro de outra universidade não tem experiência de administração e todo procedimento pode ser feito remotamente. Note que a nuvem proposta pode ser integrada a outras soluções de colaboração e geodistribuição, como federação de nuvens [Barros et al., 2015], pois a infraestrutura completa é vista com uma única nuvem OpenStack. O uso de federação, por exemplo, pode ser uma solução utilizada para aumentar o número de sítios suportados.

3. OpenStack

O OpenStack é um software de código aberto responsável pelo gerenciamento dos recursos computacionais de infraestruturas de nuvens. Essa plataforma se destaca pela sua grande comunidade desenvolvedora e por sua lista de patrocinadores, que incluem empresas como Google, Intel, IBM e HP [OpenStack, 2015]. O OpenStack possui arquitetura modular, subdividida em projetos, favorecendo sua personalização para diferentes cenários. Esses projetos são serviços especializados para cada tarefa de gerenciamento da nuvem. Neste trabalho utiliza-se a versão Juno do OpenStack.

3.1. Projetos

A arquitetura do OpenStack é dividida em projetos com diferentes finalidades. Nesta seção, são considerados apenas os projetos relacionados aos serviços necessários à uma nuvem IaaS, caso da nuvem colaborativa utilizada cuja arquitetura está ilustrada na Figura 1. Esses projetos estão ilustrados na Figura 2 e são detalhados a seguir:

- ***Dashboard***: Esse projeto, também conhecido como Horizon, fornece uma interface *web* de gerenciamento do sistema. Para tal, esse projeto se comunica por meio de APIs (*Application Programming Interfaces*). Através da interface *web* os usuários podem criar, gerenciar e acessar suas VMs, enquanto o administrador pode executar tarefas como criação e especificação dos limites de alocação de recursos computacionais para os usuários.
- ***Compute***: Também conhecido como Nova, esse serviço é responsável pela interação com o hipervisor, permitindo o provisionamento e o gerenciamento de VMs. O Nova é composto por diversos módulos que se comunicam entre si e estão distribuídos pelas máquinas da infraestrutura, como mostrado mais adiante.
- ***Image Service***: Esse projeto, também conhecido como Glance, oferece um serviço de descoberta, registro, recuperação e armazenamento de imagens para inicialização de novas VMs. Uma imagem pode ser uma nova instalação de sistema operacional ou até mesmo uma cópia do disco de uma VM já configurada. O próprio usuário pode submeter suas imagens para o repositório do Glance.

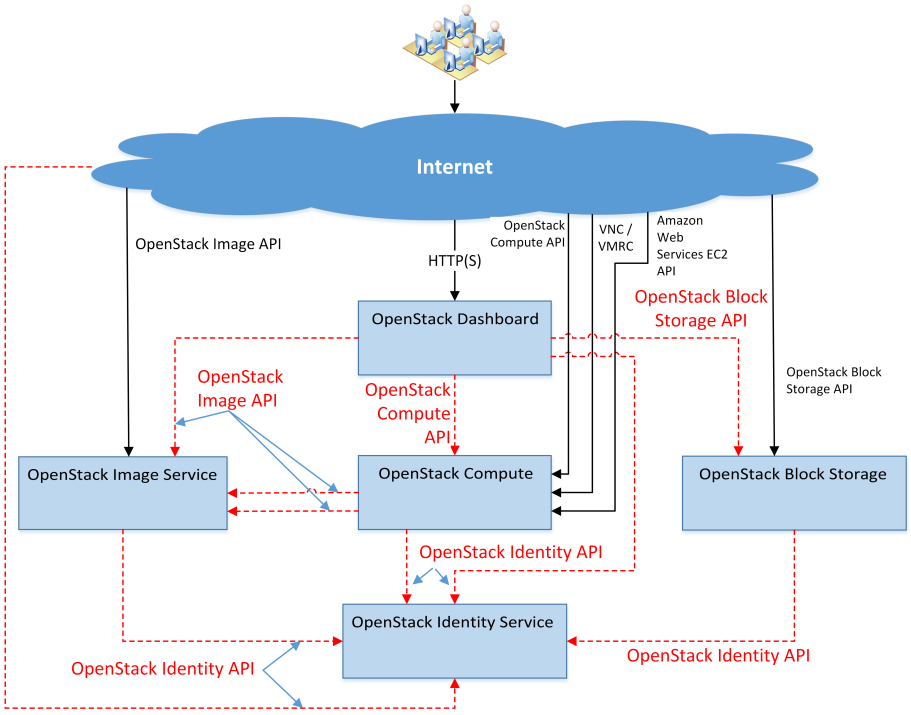


Figura 2. Interação dos projetos no OpenStack. As caixas azuis representam os projetos *Dashboard, Image Service, Compute, Block Storage* e *Identity Service*.

- ***Block Storage***: Conhecido também como Cinder, esse projeto fornece um serviço de armazenamento persistente para as VMs, em unidades denominadas volumes. Esses volumes representam discos virtuais e podem ser acessados quando estão associados às instâncias de VMs. Quando contém uma imagem, um volume pode ser uma unidade de inicialização (*boot*) para uma VM.
- ***Identity Service***: Também conhecido como Keystone, esse projeto é responsável pelo gerenciamento de identidades dos usuários e controle de acesso aos serviços.

Alguns dos projetos do OpenStack são divididos em módulos. Dentro de cada projeto, os módulos se comunicam utilizando filas de mensagens implementadas pelo *middleware* RabbitMQ [RabbitMQ, 2015], detalhado mais adiante. Além disso, um banco de dados MySQL é utilizado para armazenar informações referentes a cada projeto.

Os projetos Nova, Cinder, Glance e Keystone, todos utilizados na nuvem colaborativa deste trabalho, são exemplos de projetos do OpenStack divididos em módulos. Apesar das descrições detalhadas dos módulos serem omitidas neste trabalho por questões de concisão, a Figura 3 mostra a distribuição dos módulos entre os diversos tipos de máquina da arquitetura da Figura 1. Note que o Controlador hospeda todos os serviços relacionados a APIs, além do banco de dados e das filas de mensagens. Observe ainda que módulos responsáveis pela comunicação com hipervisores (nova-compute) ou por oferecer funcionalidades de redes para as VMs (nova-network) só existem nos nós servidores. Esses módulos não estão, por exemplo, presentes no controlador já que as VMs não são hospedadas neste nó. Por outro lado, módulos de interação com o usuário, por exemplo para acesso à VM (nova-novncproxy) ou para autenticação (nova-consoleauth e o projeto Keystone) são exclusivos do controlador. A principal diferença entre o Servidor de VMs e o Servidor de VMs e Discos é o módulo cinder-volume, utilizado para gerenciamento

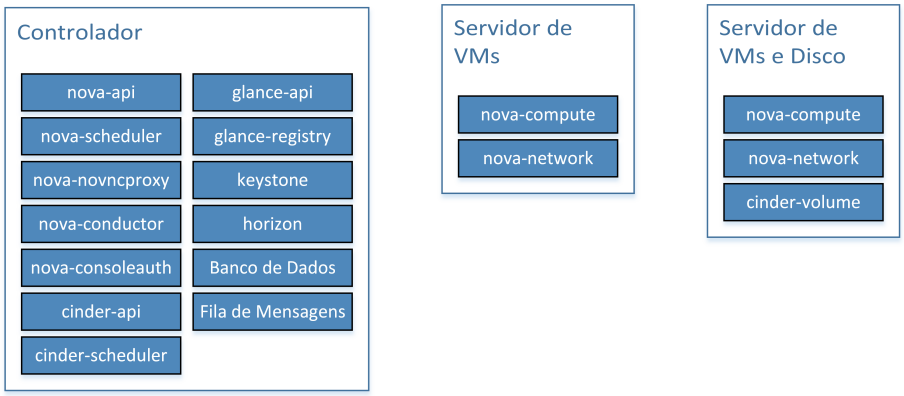
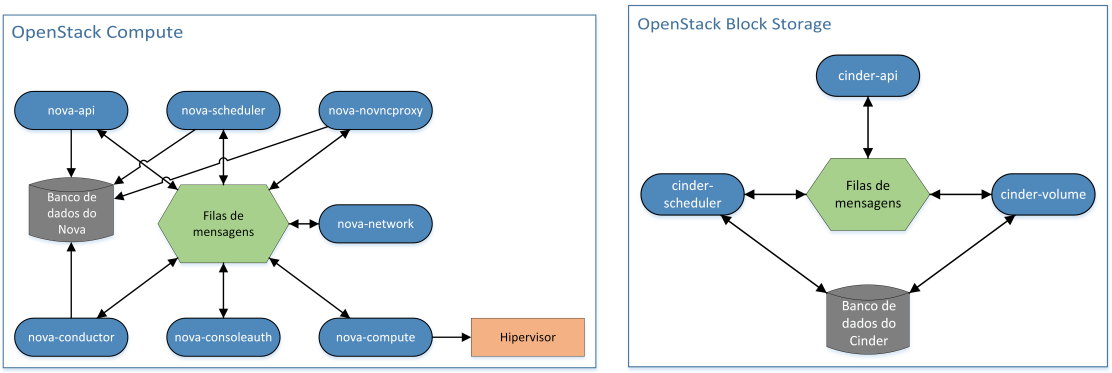


Figura 3. Distribuição dos módulos do OpenStack na arquitetura utilizada. A distribuição é realizada conforme os serviços oferecidos pela nuvem e a consequente função de cada um dos módulos na arquitetura.



(a) Componentes do Nova. (b) Componentes do Cinder.

Figura 4. Exemplo de projetos do OpenStack que utilizam comunicação direta entre os módulos e indireta com o banco de dados.

de volumes, e a hospedagem de um serviço NFS (*Network File System*), não mostrado na figura, utilizado para o armazenamento de discos virtuais.

3.2. Comunicação entre Serviços e Módulos

No OpenStack, os módulos e serviços podem se comunicar indiretamente por meio de filas de mensagens, ou diretamente por meio de APIs. Além disso, os módulos se comunicam com o banco de dados a partir de comandos MySQL. A fila de mensagens é utilizada exclusivamente para comunicação entre os módulos de um determinado projeto, como mostram as Figuras 4(a) e 4(b). Essas ilustram os módulos do Nova e do Cinder, respectivamente, e a comunicação entre eles a partir da fila de mensagens. A comunicação direta é utilizada para comunicação entre módulos de diferentes projetos e interação com o banco de dados que, na arquitetura proposta, está centralizado no Controlador.

A fila de mensagens utiliza o protocolo AMQP *(Advanced Message Queuing Protocol)* [ISO/IEC, 2014], implementado pelo *middleware* de mensagens RabbitMQ [RabbitMQ, 2015]. O AMQP é um protocolo aberto, cujo objetivo é promover a troca de mensagens entre aplicações diferentes, independentemente de como elas são implementadas. O RabbitMQ implementa um modelo *publish/subscribe* de troca de men-

Tabela 1. Configuração das máquinas utilizadas nos experimentos.

Máquina	CPU	RAM
Controlador	Intel Core i7 CPU 860 @ 2,80 GHz	8 GB
Servidor 1	Intel Core i7-4930K CPU @ 3,40 GHz	32 GB
Servidor 2	Intel Core i7 CPU 860 @ 2,80 GHz	8 GB
Servidor 3	Intel Xeon CPU E3-1241 v3 @ 3,50 GHz	32 GB
Servidor 4	Intel Core2 Quad CPU Q9400 @ 2,66 GHz	6 GB

sagens. Os módulos que geram mensagens enviam-nas para o *middleware*, que armazena as mensagens na fila apropriada. Essas mensagens ficam armazenadas até o momento em que os módulos que registraram interesse em recebê-las estejam prontos para consumi-las. Devido à característica geodistribuída da nuvem colaborativa, as mensagens geradas nos servidores são enviadas à fila RabbitMQ hospedada no Controlador.

Para comunicação direta, cada projeto do OpenStack possui uma API, que fornece um conjunto padronizado de requisições para que outras aplicações possam acessar seus serviços. Essas APIs são implementadas como serviços *web*, utilizando o padrão REST (*REpresentational State Transfer*). Dessa forma, os serviços do OpenStack podem ser acessados através da Internet, como ilustrado na Figura 2. Além de serem usadas para interação com componentes externos à infraestrutura, as APIs são utilizadas para comunicação entre módulos de projetos diferentes. Como os serviços de API são hospedados no Controlador da infraestrutura, essa máquina também centraliza o tráfego desse tipo de comunicação. Além disso, o Controlador centraliza o tráfego MySQL, já que essa máquina hospeda o banco de dados utilizado em todos os projetos.

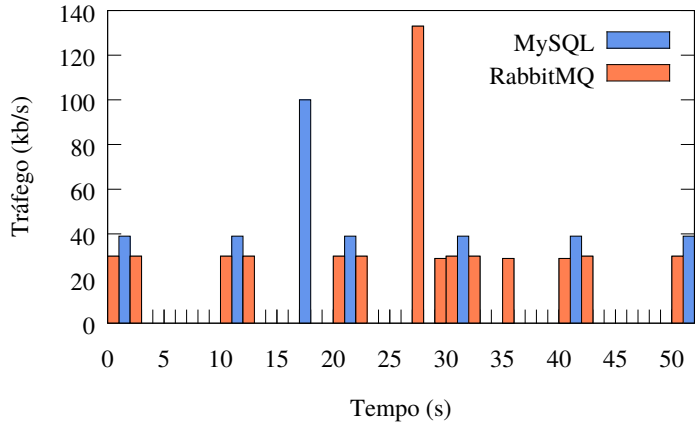
Como visto anteriormente, o Controlador é responsável por todas as interações entre os módulos, centralizando o tráfego de controle. Consequentemente, é importante estudar o tráfego gerado nessas interações, de forma a dimensionar melhor a capacidade de rede alocada ao nó Controlador.

4. Avaliação Experimental

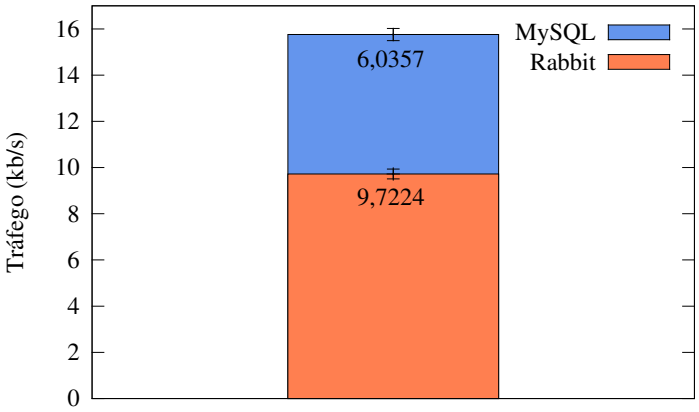
O objetivo da análise deste trabalho é avaliar o tráfego de rede entre o Controlador e os servidores sob diferentes condições. Para isso, os experimentos são realizados de acordo com a infraestrutura da Figura 1, capturando o tráfego na interface VPN do Controlador, que consiste no seu tráfego de rede WAN.

Na infraestrutura empregada nos experimentos, optou-se em utilizar somente Servidores de VMs e de Discos. Em comparação aos Servidores de VMs, esses servidores possuem um componente a mais (cinder-volume) que se comunica com o Controlador através do túnel VPN. Portanto, os Servidores de VMs e de Discos representam o pior caso para a análise proposta. Em alguns experimentos, são utilizados quatro destes servidores, caracterizando uma nuvem com quatro sítios, e em outros apenas um. Para garantir o controle sobre o cenário e isolamento de fatores externos, todos os sítios encontram-se na mesma localidade física e são conectados por uma rede local, ainda usando VPN, ao invés da Internet. A Tabela 1 contém a configuração de cada máquina da rede de testes.

Para todos os experimentos, à exceção das amostras apresentadas ao longo do



(a) Amostra do tráfego entre o Servidor de VMs e Discos e o Controlador ao longo do tempo.



(b) Distribuição do tráfego entre a comunicação dos componentes do servidor com o RabbitMQ e o MySQL.

Figura 5. Tráfego RabbitMQ e MySQL gerado a partir de um único servidor de VMs e Discos para o Controlador.

tempo, são calculadas médias e intervalos de confiança de 95%. Em algumas figuras, os intervalos não aparecem por serem muito pequenos. A partir das capturas, realizam-se diferentes análises apresentadas a seguir.

4.1. Impacto do número de Servidores de VMs e Discos

Neste experimento, inicialmente mediu-se o tráfego entre um Servidor de VMs e Discos, sem VMs instanciadas, e o Controlador. Vale notar que, neste experimento, nenhuma VM está instanciada nos servidores. A Figura 5(a) apresenta o tráfego gerado devido à troca de mensagens entre o RabbitMQ e os componentes nova-compute e nova-network e entre o banco de dados e o cinder-volume. Na infraestrutura utilizada, os componentes nova-compute e nova-network presentes nos Servidores de VMs e Discos enviam periodicamente atualizações para o nova-conductor, que executa no Controlador, utilizando o sistema de filas de mensagens do RabbitMQ. O nova-conductor então insere essas informações no banco de dados do Nova. Da mesma forma, o componente cinder-volume, presente em cada Servidor de VMs e Discos, envia periodicamente atualizações

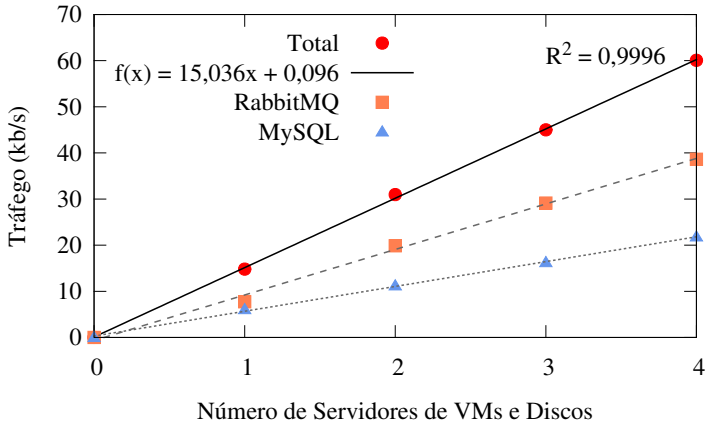


Figura 6. Tráfego na rede com o aumento do número de Servidores de VMs e Discos.

ao Controlador. No entanto, diferentemente do Nova, ele se comunica diretamente com o banco de dados do Cinder usando MySQL.

Observa-se na Figura 5(a) que o RabbitMQ se comunica com os componentes do servidor a cada 10 s, para atualizar o estado dos serviços, e que entre 20 e 30 s há um pico, correspondente à atualização da lista de instâncias dos servidores (essa atualização é realizada a cada 60 s). Da mesma forma, o cinder-volume (rótulo MySQL na figura) relata o seu estado a cada 10 s e entre 10 e 20 s há um pico, relativo à atualização das informações dos volumes (realizada periodicamente a cada 60 s). A Figura 5(b) mostra a contribuição desses dois tipos de comunicação para o tráfego médio na rede, calculado durante 60 s, em 25 experimentos diferentes, quando apenas o Servidor 1 está conectado ao Controlador. Pode-se observar que a maior parte do tráfego envolve o RabbitMQ, ou seja, o tráfego gerado pelo Nova. Além disso, é possível verificar que o tráfego total é de aproximadamente 15 kb/s.

Em seguida, incrementou-se o número de Servidores de VMs e Discos conectados ao Controlador, medindo o impacto no tráfego. A cada Servidor de VMs e Discos adicionado à infraestrutura amostrou-se o tráfego por 60 s. Este procedimento foi repetido 10 vezes. A Figura 6 apresenta o tráfego total na rede e também os tráfegos relacionados somente ao RabbitMQ e ao MySQL, em função do número de Servidores de VMs e Discos. Pode-se observar que o tráfego na rede aumenta com o crescimento do número de servidores. Em função da tendência linear observada nos resultados, foram realizados ajustes lineares dos pontos dos tráfegos que resultaram nas retas apresentadas na mesma figura. Para esse ajuste linear, o valor de R^2 indica o quão ajustado aos pontos é o modelo linearizado. Os valores de R^2 variam de 0 a 1, sendo o modelo exatamente linear para R^2 igual a 1. Para o tráfego total na rede, o valor encontrado para R^2 foi 0,9966. A partir da equação da reta obtida é possível concluir que cada servidor adiciona em média um tráfego de 15 kb/s à rede. Essa constatação coincide com o resultado da Figura 5(b). Assim, por extrapolação, em um cenário com 100 servidores, um tráfego de 1,5 Mb/s passaria pela interface de rede do Controlador.

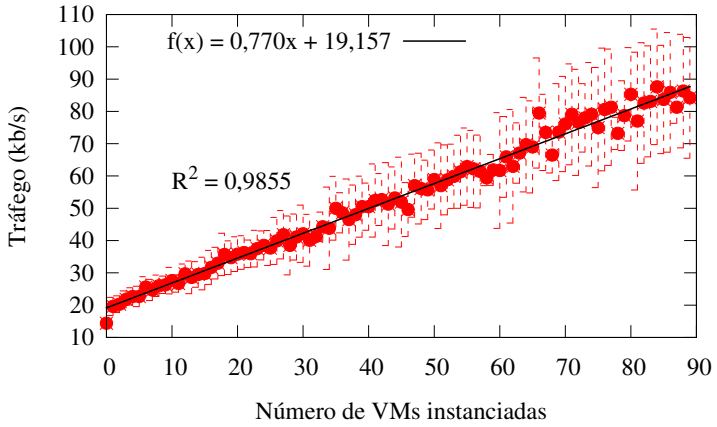


Figura 7. Tráfego na rede com o aumento do número de VMs.

4.2. Impacto do número de VMs por Servidor de VMs e Discos

Neste experimento, variou-se o número de VMs instanciadas em um único Servidor de VMs e Discos (Servidor 1) para analisar o impacto no tráfego da rede. À medida que o número de instâncias de VMs e volumes aumenta, mais dados precisam ser enviados para o Controlador para atualizar os bancos de dados do Nova e do Cinder. Para avaliar o impacto do aumento do número de VMs instanciadas na rede, captura-se o tráfego gerado após a criação bem-sucedida de cada VM, ou seja, sem considerar o tráfego relativo ao processo de criação de VMs. Para cada número de VMs instanciadas, o tráfego é medido por 60 s. Esse experimento foi repetido 10 vezes para cada número de VMs. As VMs utilizadas no experimento e no restante deste trabalho possuem o sistema operacional CirrOS, 64 GB de RAM e 1 CPU virtual. Vale notar que as mensagens de controle não variam de acordo com a configuração das VMs e número de CPUs, já que as mensagens enviadas são informações genéricas como estado das VMs, id das VMs, etc. Consequentemente, a configuração das VMs não altera o comportamento observado nos experimentos a seguir.

A Figura 7 ilustra o resultado do experimento. Pode-se observar um comportamento aproximadamente linear, apesar da maior variabilidade do experimento conforme o número de VMs instanciadas aumenta. O ajuste linear dos dados para obter a equação da reta gerou um R^2 igual a 0,9855. Pela equação obtida, estima-se que cada VM gere uma taxa média de 0,77 kb/s. Em um cenário com 100 servidores contendo 15 VMs cada, totalizando 1500 VMs, a carga total gerada pelas VMs seria de 1,155 Mb/s. Neste cenário, somando-se a carga dessas VMs com a dos Servidores de VMs e Discos, analisada anteriormente na Seção 4.1, tem-se uma carga total de aproximadamente 2,7 Mb/s.

4.3. Impacto da criação e destruição de múltiplas VMs

Os experimentos desta seção mostram o comportamento do tráfego durante os processos de criação e de destruição de VMs. No OpenStack, é possível inicializar uma instância a partir de uma imagem armazenada em uma unidade de disco efêmera, que somente armazena os dados enquanto a instância associada existir; ou a partir de uma imagem armazenada em um volume do Cinder, que oferece armazenamento persistente. Quando o usuário solicita a inicialização de uma instância sem volume pela primeira vez, o Glance envia a imagem ao Servidor de VMs e Discos através do túnel VPN. A imagem é armazenada localmente no servidor e, então, copiada para uma unidade de disco

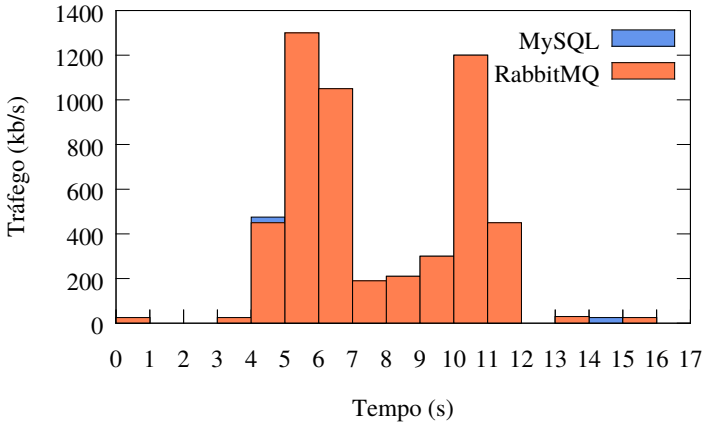


Figura 8. Tráfego gerado pela criação e destruição de uma VM.

efêmera, também armazenada no servidor. Neste caso, se uma nova instância de máquina virtual com a mesma imagem for criada no mesmo Servidor de VMs e Discos, a imagem não precisará ser enviada novamente pela rede. No caso de instâncias com inicialização a partir de volume, primeiramente um volume vazio é criado, depois a imagem é transferida para o Servidor de VMs e Discos e copiada para o volume gerenciado pelo Cinder. Diferentemente do Nova, o Cinder não possui uma política de cache [Cinder, 2015]. Consequentemente, sempre que uma nova instância com inicialização a partir de volume é criada, a imagem é transferida pela rede.

O caso mais comum de criação de VMs em uma nuvem IaaS é a partir de disco efêmero (isto é, baseada em imagem), devido ao alto tráfego gerado na criação de VMs por volume. Portanto, os casos de uso do sistema devem ser pensados de forma a restringir a utilização de instâncias inicializadas a partir de volume de forma a não sobrecarregar a rede. Para avaliar apenas as mensagens de controle geradas, nesta seção analisa-se o tráfego de rede durante a criação e a destruição consecutivas de VMs baseadas em imagem. O tráfego foi avaliado após a existência de uma imagem de VM no cache do Glance. Dessa forma, os experimentos não são dependentes da configuração utilizada para as VMs. Primeiramente, é mostrado na Figura 8 o impacto no tráfego da rede quando um usuário realiza uma requisição para criar uma instância e, ao fim desse processo, realiza outra requisição para destruí-la. Note que o tráfego atinge um valor de pico de aproximadamente 1,3 Mb/s durante a criação da VM e 1,2 Mb/s na destruição.

Para analisar o tráfego gerado em mais detalhes, emprega-se um *benchmark* para diferentes situações de criação e destruição de VMs no Servidor 1. Dessa forma, utiliza-se a ferramenta Rally [Rally, 2015], que foi especialmente desenvolvida para a realização de testes de escalabilidade e desempenho em nuvens OpenStack. Essa ferramenta oferece uma série de cenários de testes pré-definidos que simulam diferentes cargas na nuvem, sendo utilizada em trabalhos relacionados [Gelbukh, 2014, Cisco, 2014]. Neste experimento utiliza-se o cenário `boot-and-delete.json`, um dos mais simples presentes no Rally, para executar experimentos de criação e destruição de VMs e, a partir deles, calcular a média do tráfego gerado. Nesse cenário é definido um número de VMs a serem requisitadas à nuvem. O Rally gera uma requisição para criar uma instância, espera esta ser criada, e depois solicita sua destruição. Na sequência, faz um novo ciclo até atingir o

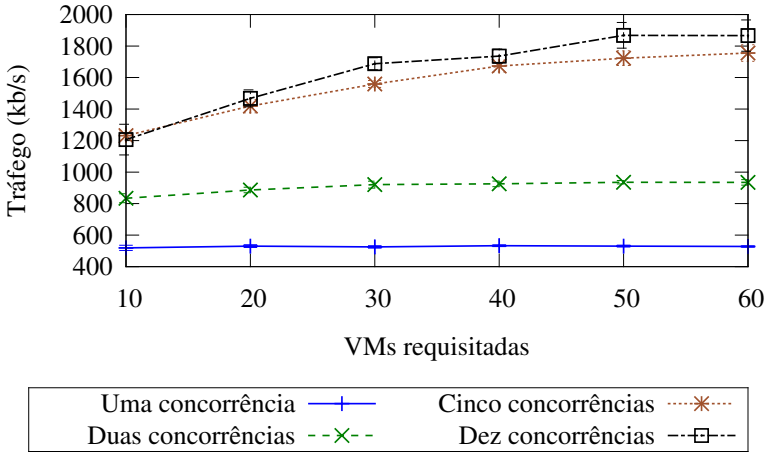


Figura 9. Tráfego gerado nas criações e destruições concorrentes de VMs.

total de instâncias requisitadas. Além disso, o Rally permite configurar a quantidade de requisições paralelas que são enviadas ao Nova API, representando as ações concorrentes de múltiplos usuários da nuvem. Para um valor de concorrência igual a cinco, por exemplo, são emitidas cinco requisições paralelas por vez e, à medida que uma acaba, uma nova é enviada para manter o Nova API ocupado com um valor constante de requisições em paralelo. Nos experimentos realizados foram criadas VMs utilizando-se valores de concorrência de 1, 2, 5 e 10. O experimento foi repetido 10 vezes para cada número de VMs requisitadas e concorrências.

A Figura 9 mostra o tráfego gerado. Note que, mesmo para um número pequeno de usuários concorrentes considerado nos experimentos, o tráfego pode ser elevado. Por exemplo, para dez usuários executando o ciclo com dez VMs, o tráfego médio gerado durante o processo é de 1,2 Mb/s. Isso pode representar um alto impacto na rede para uma nuvem de pequeno porte, considerando também o tráfego já gerado continuamente pelas mensagens periódicas, como visto nas Seções 4.1 e 4.2. Ainda na Figura 9, observa-se que a variação de uma concorrência para duas concorrências faz o tráfego médio aproximadamente dobrar. No entanto, o tráfego gerado para cinco concorrências não é cinco vezes maior do que o tráfego com uma concorrência. Além disso, de cinco para dez concorrências, o tráfego praticamente não se altera, pois Servidor de VMs e Discos utilizado não suporta a criação paralela de todas as VMs requisitadas.

5. Trabalhos Relacionados

Alguns trabalhos da literatura abordam a escalabilidade de componentes do OpenStack. Gelbukh, em seu trabalho, avalia a escalabilidade de uma versão do OpenStack desenvolvida pela Mirantis, que é uma das maiores contribuidoras para o código-fonte do OpenStack [Gelbukh, 2014]. Mais especificamente, o trabalho avalia a quantidade de requisições simultâneas para a criação de VMs que uma infraestrutura OpenStack consegue atender. Os resultados mostram que a infraestrutura utilizada nesse trabalho consegue servir 250 requisições paralelas de criação de VMs. Além disso, foi possível instanciar 75.000 VMs na infraestrutura estudada. Apesar de os resultados serem dependentes do *hardware* dos servidores e da rede empregada na infraestrutura, os resultados mostram que é possível alcançar um alto nível de escalabilidade no OpenStack.

A Cisco também realiza diversos experimentos para estressar os componentes do OpenStack e determinar, por exemplo, qual é a limitação em relação ao número de servidores de VMs que um único nó controlador suporta e qual o número máximo de VMs que podem ser hospedadas em cada servidor [Cisco, 2014]. Os resultados mostram que o número de servidores de VMs é limitado pelo RabbitMQ, que consiste na solução de comunicação entre o controlador e os servidores detalhada na Seção 3.2. Além disso, os resultados mostram que o tempo necessário para criar uma VM aumenta de acordo com o número de VMs já instanciadas na infraestrutura, mesmo se essas VMs estiverem em repouso. Devido a esse comportamento e a outros fatores apontados no trabalho, o número de VMs que podem ser instanciadas é limitado, mesmo se ainda existir memória RAM disponível para sua criação. O estudo recomenda, empiricamente, utilizar em cada servidor apenas 40% de sua capacidade teórica de hospedagem de VMs, calculada em função da capacidade de memória RAM do servidor.

Apesar de os trabalhos citados anteriormente consistirem em experimentos em massa, estressando diversos componentes do OpenStack, esses não realizam uma análise da sobrecarga de rede gerada por mensagens de controle. Dessa forma, devido à importância dessa análise para nuvens geodistribuídas, este trabalho complementa a literatura, estudando a escalabilidade do OpenStack do ponto de vista da capacidade de rede.

6. Conclusões e Trabalhos Futuros

Este trabalho analisou impacto do tráfego de controle em uma nuvem colaborativa com controlador centralizado, que por sua vez se conecta a servidores de VMs através de uma WAN. Esse cenário reproduz uma configuração típica de uma nuvem colaborativa de pequeno porte. Os experimentos mostraram que, mesmo com as VMs em repouso, mensagens são continuamente trocadas pela WAN entre o controlador e os servidores de VMs. Na arquitetura IaaS considerada, cada comunicação entre um servidor e o controlador gera um tráfego médio de aproximadamente 15 kb/s. Além disso, quanto maior for o número de VMs instanciadas, maior é o tráfego para o controlador, mesmo se as VMs não realizarem tarefas de rede. Isso ocorre pois o servidor de VMs relata periodicamente os estados de suas VMs para o controlador. Os experimentos mostraram que cada VM adicionada gera, em repouso, um tráfego médio de aproximadamente 0,77 kb/s. Apesar de os valores individuais serem baixos, em uma nuvem IaaS de tamanho médio, com 100 servidores e 15 VMs por servidor, esses valores produzirão um tráfego médio, de controle, de 2,7 Mb/s. Esse resultado é agravado quando considera-se o aumento de carga na rede durante atividades de criação e destruição de máquinas virtuais. Por exemplo, a criação e posterior destruição de 10 VMs em paralelo, cada uma por um usuário diferente, gera um tráfego médio de controle de 1,2 Mb/s durante o processo. Assim, conclui-se que o tráfego de controle gerado deve ser absolutamente considerado no projeto de uma nuvem colaborativa geodistribuída, e que este deve ser estimado de acordo com o perfil esperado de utilização de máquinas virtuais pelos usuários.

Como trabalhos futuros, pretende-se a realização de experimentos considerando as VMs em uso, por exemplo, no caso de suas aplicações instaladas gerarem determinado padrão de tráfego na rede. Além disso, é possível estender a análise considerando a existência de outros projetos do OpenStack, como o Neutron (serviço avançado de rede) e o Ceilometer (serviço de monitoramento de recursos), o que aumentará o tráfego de controle. Por fim, como as redes WANs possuem enlaces com alta latência devido às

distâncias geográficas e possíveis períodos de congestionamento, é importante verificar o impacto da latência da rede nos serviços do OpenStack.

Referências

Apache (2015). *Apache CloudStack*. The Apache Software Foundation. http://cloudstack.apache.org - Acessado em Dezembro de 2015.

Barros, A., Brasileiro, F., Farias, G., Germano, F., Nóbrega, M., Ribeiro, A. C., Silva, I. e Teixeira, L. (2015). Using fogbow to federate private clouds. Em *Salão de Ferramentas do XXXIII SBRC*.

Cinder (2015). *Generic image cache functionality*. OpenStack Cinder Team. http://specs.openstack.org/openstack/cinder-specs/specs/liberty/image-volume-cache.html - Acessado em Dezembro de 2015.

Cisco (2014). *OpenStack Havana Scalability Testing*. Cisco Systems. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/OpenStack/Scalability/OHS.pdf - Acessado em Dezembro de 2015.

Couto, R. S., Sciammarella, T., Barreto, H. F. S. S. M., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M. K., Vetter, F. e Marins, A. L. A. (2015). GT-PID: Uma nuvem IaaS universitária geograficamente distribuída. Em *Quinta Conferencia de Directores de Tecnología de Información - TICAL 2015*, p. 1–19. Redclara.

Develder, C., De Leenheer, M., Dhoedt, B., Pickavet, M., Colle, D., De Turck, F. e Demeester, P. (2012). Optical networks for grid and cloud computing applications. *Proceedings of the IEEE*, 100(5):1149–1167.

Endo, P. T., de Almeida Palhares, A. V., Pereira, N. N., Gonçalves, G. E., Sadok, D., Kelner, J., Melander, B. e Mangs, J. (2011). Resource allocation for distributed cloud: concepts and research challenges. *IEEE Networks*, 25(4):42–46.

Gelbukh, O. (2014). *Benchmarking OpenStack at megascale: How we tested Mirantis OpenStack at SoftLayer*. Mirantis. http://www.mirantis.com/blog/benchmarking-openstack-megascale-tested-mirantis-openstack-softlayer/ - Acessado em Dezembro de 2015.

ISO/IEC (2014). *ISO/IEC 19464. Information technology – Advanced Message Queuing Protocol (AMQP) v1.0 specification*. ISO/IEC.

Jennings, B. e Stadler, R. (2015). Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, 23(3):567–619.

Kivity, A., Kamay, Y., Laor, D., Lublin, U. e Liguori, A. (2007). KVM: the linux virtual machine monitor. Em *Linux Symposium*, volume 1, p. 225–230.

OpenStack (2015). *OpenStack - Cloud Software*. OpenStack Foundation. http://www.openstack.org - Acessado em Dezembro de 2015.

RabbitMQ (2015). *RabbitMQ - Messaging that just works*. Pivotal Software. https://www.rabbitmq.com/ - Acessado em Dezembro de 2015.

Rally (2015). *Rally*. Rally Team. http://wiki.openstack.org/wiki/Rally - Acessado em Dezembro de 2015.

A Middleware for Data-centric and Dynamic Distributed Complex Event Processing for IoT Real-time Analytics in the Cloud

Gustavo B. Baptista, Felipe Carvalho, Sergio Colcher and Markus Endler

Department of Informatics – Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil.

{gbaptista,fcarvalho,colcher,endler}@inf.puc-rio.br

Abstract. *IoT big data real-time analytics systems need to effectively process and manage massive amounts of data from streams produced by distributed data sources. There are many challenges in deploying and managing processing logic at execution time in those systems, especially when 24x7 availability is required. Aiming to address those challenges, we have developed and tested a middleware for Distributed CEP, with a data-centric and dynamic design, based on the Data Distribution Service for Real-Time Systems (OMG-DDS) specification and its extension for dynamic topics/types (DDS-XTypes). Its main advantages include the use of OMG-DDS; witch is suitable for IoT applications with QoS requirements, its dynamic capabilities, and the scalable and parallel execution of the CEP rules on a dynamic set of processing nodes.*

1. Introduction

Processing data streams is one of the main challenges facing today's distributed and pervasive systems, given the high dynamicity, heterogeneity and volume of real-time data generated by contemporary network-centric systems. Emerging Internet of Things (IoT) [Greengard 2015], Big Data [Nathan and Warren 2015] and Real-time Data-Stream Analytics [Byron 2014] applications (e.g. smart cities, smart energy grids, bio-surveillance, environmental monitoring, fleet monitoring and smart manufacturing) need to effectively process and manage massive amounts of data from streams produced by distributed data sources, in order to perform real-time analytics for supporting timely decision-making and timely response. Other application domains of real-time analytics include social networks, stock market analysis, IT infrastructure monitoring, etc. Data streams are infinite flows of data generated by wide array of fixed and mobile sensors or devices used in modern pervasive/mobile IoT applications. However, there are challenges in deploying and managing processing logic for these data streams in a distributed manner on datacenters or clouds [Erl et al. 2013]. In particular, there is usually a high operational cost involved in setting up a distributed data stream processing environment or dynamically modifying its processing logic without stopping the whole system, specially for stream analysis systems demanding 24x7 availability. Such dynamic changes may become necessary to accommodate variant needs from the application domain, such as inclusion of new patterns of data/events to be detected, or technology upgrades, i.e. changes of the basic data types of the input streams due to new types of sensors. Among data stream processing techniques, Complex Event Processing (CEP) [Luckham 2001] is one of the most complete models. It views data items as notifications of events from multiple sources, and uses rules for describing patterns of low-level events, that, when detected, produce higher-level events whose occurrence are

notified to the interested parties. In many CEP systems, the processing logic is described as a network of event processing agents (EPAs), or EPN. In a Distributed CEP system (DCEP), an EPN can be distributed with EPAs deployed across many different machines [Etzion and Niblett 2010]. Aiming to address the challenges of setting up and dynamically managing the deployment of mechanisms for CEP-based data stream processing, we have developed and tested a middleware for DCEP, based on the Data Distribution Service for Real-Time Systems (DDS) [OMG 2006] specification and its extension for dynamic topics/types (DDS-XTypes) [OMG 2014], to support on-line detection of patterns. Its main advantages include the use of DDS; witch is suitable for IoT applications with QoS requirements, its dynamic capabilities, and the scalable and parallel execution of the CEP rules on a dynamic set of processing nodes on a datacenter or in the cloud. The middleware uses the RTI Connext DDS [RTI 2016] as its communication backbone.

2. Data-Centric Paradigm

In distributed systems, different processes need to manage and exchange data. In the same way as with data management and persistence, communication middleware can be designed around different entities, which we refer to as different communication paradigms. This paradigm influences the design of the middleware clients (e.g. services, applications or other systems), their functional capabilities and compliance to non-functional requirements. In a *message-centric* communication paradigm, the unit of information exchange is the message where the main focus is the delivery of messages to destinations, without deep knowledge of message's structure and contents, i.e. data is sent as an opaque sequence of bytes (the payload inside messages), and serialization is delegated to the application. Many communication technologies follow this approach, ranging from simple point-to-point connections to more advanced publish/subscribe communication infrastructures, such as, Java Message Service (JMS) [Oracle 2016a], Advanced Message Queuing Protocol (AMPQ) [OASIS 2012]. Some message-centric systems, known as content-based publish/subscribe systems, such as PADRES [Fidler et al. 2005], have mechanisms that enhance the communication model to be able to access the contents of messages, allowing the routing of messages by defining expressions of interest based on event properties. Another similar paradigm is the *log-centric* approach offered by systems such as Kafka [Apache Software Foundation 2016], where data topics are managed as append-only logs, which can be partitioned among different Kafka brokers for parallelism. With a *data-centric* communication paradigm [Schneider 2012], the mean of interaction is data. The middleware infrastructure has the definition of the structure of the data it manages and it is fully aware of the contents (i.e. values) of these structures, and imposes rules on how data is structured, changed and accessed (i.e. how participant nodes receive updates). While in this model the communication is peer-to-peer, there is a notion of a *global shared data space*, which is decentralized and maintained by all peers and contains the structure and instances of data, and nodes access this space in order to read and write information. The role of the infrastructure is to ensure that all participant nodes have a consistent and up-to-date view of this data space. Two main aspects that can be influenced by those different paradigms are how distributed state is managed and how it is communicated among nodes of a distributed system. The management of distributed state is a fundamental design decision, and the communication paradigm will have a direct relation on the type of interaction between distributed nodes and with the state sharing and management approach. State

management in message-centric systems has to be performed by applications since the middleware is not aware of the messages contents. Since with a data-centric approach messages are not sent about data, but data itself is kept in a global data space, state is inherently disseminated and managed (i.e. reflected into the global data space) by the infrastructure. The infrastructure has the definition of data schemas, and the instances of these schemas are kept as globally accessible objects. In the same way as in relational databases, data-centric middleware provide a fixed set of operations over the global data space, with an important and significant impact on reducing systems integration efforts [Schneider 2012; Tambe 2012]. In order to realize the data-centric paradigm, a middleware implementation following this approach is necessary. In the next section, we summarize the OMG DDS specification, which is a standard for real-time data-centric publish/subscribe middleware, and a few implementations of this specification.

2.1. Data Distribution Service for Real-Time Systems (OMG-DDS)

The Data Distribution Service for Real-Time Systems (DDS) [OMG 2006] standard defines a fully distributed peer-to-peer (i.e. broker-less) real-time data-centric publish/subscribe communication model. It provides high performance communication, scalability and availability, and supports the specification of Quality of Service (QoS) contracts between data producers and consumers, and also mechanisms for dealing with real-time aspects (e.g. priority and other specific QoS policies). It allows interoperability across different DDS implementations, programming languages and platforms, as well as automatic discovery of DDS publishers/subscribers. DDS Topics are first class entities for information transfer, which application peers can publish or subscribe to, and can be regarded as distributed relational database tables. Topics are defined by the application programmer, which writes an IDL file describing the names, data types, and the keys that identify the instances (that can be seen as database rows) of each Topic, to which updates are called *samples*. A compiler takes the IDL as input, and generates code for communication stubs in a desired programming language. By using the generated stubs, applications can join a DDS Domain and publish and subscribe to data in this Domain. The DDS Domain, which contains the global shared data space, is fully distributed over the peer-to-peer network formed by participating network nodes, without any intermediate broker or centralized management entity. DDS applications produce and consume data to this global data space without any direct knowledge of the parties involved in the production and consumption of data (i.e. naming and location transparency). The DDS specification provides a large set of QoS features and enforcement mechanisms. Every producer and consumer of data defines its QoS requirements or capabilities through QoS contracts that are matched by a protocol, called Requested-versus-Offered protocol, which checks QoS matching before any establishment of communication. Those QoS parameters address many aspects, such as communication reliability, latency or transport priority, data persistence, and behavior of node discovery mechanisms, among many others.

3. Data-Centric and Dynamic Design Approach

In DCEP systems, we will herein refer to the mechanisms responsible for disseminating data as the *communication layer* (communication middleware, data objects, marshaling, etc), and the mechanisms responsible for processing the data (e.g. event processing agents (EPAs) and event processing networks (EPNs), CEP engines, event types, etc) as

the *processing layer*. There are scenarios where knowing all event types and situations (i.e. CEP rules in EPAs) to be detected beforehand is not possible. Teams responsible for monitoring activities must be able to define those events and situations on the fly, to support uninterrupted monitoring. In a dynamic scenario, the DCEP system processing and communication layers must be capable of dealing with the definition of those logical entities dynamically, and keep those layers consistent with each other. Our dynamic and data-centric approach aims at achieving a fully dynamic and integrated environment for DCEP entities and event types both at the processing and communication layers.

3.1. Global Reactive Data Space

In order to do so, in this work, the concept of the *global shared space* of the DDS specification is extended to include not only topics that implement the actual transmission of data, but also the modeling of processing entities, i.e. event types and DCEP entities, in a data-centric way. By doing so, dynamic capabilities are provided in a way that promote interoperability and reduce complexity in accessing and managing the DCEP system, while obtaining the other benefits of using DDS for the dissemination of actual events, such as shared state management and consistency, decoupling, availability, scalability and performance. We propose that in addition to the DDS Topics that perform the actual transmission of data, this data space should be used to keep the description of the entities of the DCEP system. The global data space would keep two additional mechanisms: a global dictionary of event types that can be agreed upon by all participants in a communication, and the definition of DCEP entities, such as EPAs, EPNs, CEP Rules, etc and deployment information for these entities in any desired processing node. Thus, the data dissemination and reactive behavior over that same data are modeled together inside the global data space. By extending the global data space with reactive behavior, we call it the *global reactive data space*. The dynamic data-centric solution has to be implemented by three main middleware services: one for the definition of event types, other for the generation and management of dynamic DDS topics, and finally, the processing layer of the DCEP system that uses the above services, which allow the definition of DCEP entities and deployment information for those entities. While on a static approach the mapping of event types in the processing and communication layers would had to be performed by external mechanisms, that generate and keep track of static DDS topics for each existing event type in the processing layer, with our approach, when an event type is defined, both processing and communication layers have the same description of it, and a dynamic DDS topic is created for this type automatically. In order to do so, we make use of the extension specification for Dynamic DDS Topics, called DDS-XTypes [OMG 2014], to which RTI Connext DDS [RTI 2016] provides an implementation for. In addition to all the benefits obtained from using data-centric middleware for the dissemination of events, as mentioned in the previous section, the modeling of reactive behavior inside the global shared space allows dynamic capabilities to be fully supported without the use of many external dynamic allocation mechanisms, bringing benefits in terms of increased decoupling and interoperability, reduced complexity in management, evolution and maintainability. By using this approach, any data consumer, producer or EPA, can be defined in a dynamic way, only by specifying the data type it produces or consumes, and start writing or listening for updates on that type. There is no need to define any software module implementing the data type: it is defined or retrieved from

the metadata service in the global data space. This facilitates the dynamic deployment of producers, consumers and especially of DCEP entities. For example, a complete EPN implementing a processing logic could be defined at runtime, only requiring the specification of the event types to be written into the metadata service, and then the definition of EPAs and their rules referencing those types. Since the event types referenced by the CEP rules are in the same global data space, and all have correspondent dynamic DDS topics associated to them, the dynamic deployment is seamless.

4. D3CEP Middleware

D3CEP (Data-Centric Dynamic Distributed CEP) is a DCEP middleware/system, which follows a clustered distribution model. It allows the deployment of Event Processing Agents (EPAs) on distributed nodes tightly coupled in a computer network in a cluster of physical machines or cloud computing instances. Those agents are logically connected to each other to form Event Processing Networks (EPNs) that compose global processing logics to detect situations. This clustered distributed model aims to provide high throughput and low latency among processing nodes, and to provide scalability in the number of event producers, consumers, and situations to be detected [Cugola and Margara 2012].

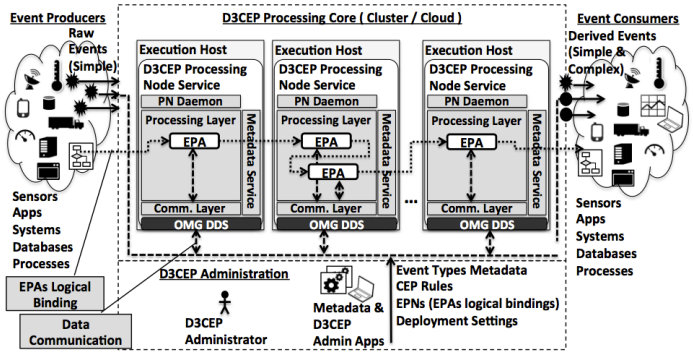


Figure 1. D3CEP Solution Overview

D3CEP uses a data-centric paradigm for the communication and the processing of events. In order to achieve data-centric communication, the D3CEP Communication Layer implements a middleware API using a Data-Centric Publish-Subscribe (DCPS) model provided by an implementation of the OMG DDS [OMG 2006] specification. The use of this peer-to-peer DCPS communication model aims to provide an optimized performance for the routing of events among processing nodes that execute EPAs. The middleware also handles the management of state among EPAs for the processing of distributed CEP rules, so EPAs, event producers and consumers don't need to perform this management. These features promote a reduction in the coupling among all entities (e.g. producers, consumers, rules, etc). Furthermore, in accordance to a data-centric approach for communication, the D3CEP Processing Layer uses a data-centric CEP engine implementation [EsperTech 2016], at each of the processing nodes, with CEP rules deployed inside EPAs, which in turn are deployed on those engines. In order to meet the requirements of dynamic and flexible behavior, D3CEP provides flexibility in the definition and re-definition of event types, rules and EPNs at execution time, without requiring stopping the system, event producers or consumers. Figure 1 depicts an example of the main elements of the solution, shown with a typical organization of a distributed complex event processing system, with event producers, which input raw

events into the system, a processing core composed by a set of distributed processing nodes, where EPNs can be composed by the deployment of EPAs, and event consumers, which subscribe to receive derived events from the system. EPNs are represented in the figure by the thin dashed lines connecting all EPAs, which are logically bounded to each other, while the bold dashed lines connecting the execution hosts represent the actual data communication between the EPAs.

4.1. Middleware Layers

The main middleware layers of the D3CEP solution are the Processing Node Daemon, Metadata Sharing Layer, Processing Layer and the Communication Layer. Figure 1, presented previously, depicts an example of the middleware layers deployed on execution hosts, inside each node's PN Service instance. Figure 2 details the layers and their main internal components. The ***Processing Node Daemon*** periodically collects the node's information and publishes a heartbeat event inside the D3CEP domain, indicating the nodes current status and availability for the deployment of EPAs. This information can be consumed, for example, by an administration application (or an automatic deployment mechanism) for information useful for the deployment of EPAs.

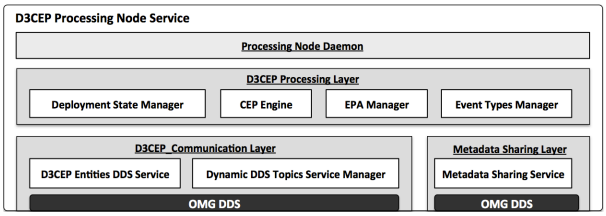


Figure 2. The D3CEP Middleware Layers.

In order to provide dynamic capabilities, the ***Metadata Sharing Layer*** provides mechanisms for the dynamic definition of metadata. The ***Metadata Sharing Service*** is a DDS-based service that allows the sharing of data types descriptions. Therefore, it provides peers in a DDS domain with a view of all existing data types descriptions in this shared data space, so that applications can query them or subscribe to receive updates. It provides an API that facilitates the development of applications that access a shared space with metadata descriptions. The ***Metadata*** element is a logical structure that describes a data type. A Metadata instance has a name and a set of members. Each member has a name and a type (e.g. string, integer, etc). For example, a metadata instance could be named as "Location", and composed by members named as "latitude" and "longitude" both of the type "double". The service is used in D3CEP as the global catalog of event types, which defines the event types to be exchanged by the Communication Layer, as to be processed by the Processing Layer. The administrator of the system can define the event types through the Metadata Administrator application, which uses the Metadata Sharing Service to publish Metadata instances into the DDS domain. When an event type is defined by the administrator, all processing nodes, event producers and consumers that have an instance of the service running on the same domain will have a view of the existing event types.

At the base of the D3CEP solution is the ***Communication Layer***, which implements data dissemination inside the D3CEP core, implementing middleware APIs on top of the OMG DDS specification. This layer is used for two main purposes: disseminating D3CEP entities, such as EPNs, EPAs, deployment settings, etc, and simple and complex events from event producers, consumers and among processing

nodes. Therefore, two main services are used inside this layer, the D3CEP Entities DDS Service and the Dynamic DDS Topics Service, which are explained below. The ***D3CEP Entities DDS Service*** is a DDS service used inside the D3CEP core, which implements the sharing of entities that represent D3CEP management information, allowing an administration application to define CEP entities, such as EPNs and EPAs and their deployment. The system administrator defines the management information and publishes it into the domain. Processing nodes are passive entities, and react deploying entities locally when they are published. Therefore, there is a concept of a *Deployment State*, which is a snapshot of the system current configuration state. A Deployment State contains a set of EPNs defined, each of them composed by a set of EPAs. Also inside a Deployment State instance, there are Processing Nodes Deployments, which are a mapping of the existing EPAs to Processing Nodes. The Processing Layer on the correspondent processing nodes defined by this mapping will deploy these EPAs locally, when the Deployment State is received in those nodes. The ***Dynamic DDS Topics Service*** is a middleware API that facilitates the development of DDS applications and services that need dynamic DDS topics described at runtime. It provides a way of creating dynamic DDS topics by providing a simple data description (the aforementioned Metadata class). It is used in D3CEP to allow a data-centric and dynamic dissemination of simple and complex events from event producers, consumers and event processing agents. The CEP rules in D3CEP are dynamically defined, and dynamically deployed in distributed processing nodes to receive events as input and to produce events as output. The event types referenced in those rules need to have a mapping to underlying DDS topics for the communication to take place. However, due to this dynamic aspect, the topics cannot be defined at development time, to be statically generated. Therefore, this service is used to implement the creation of such dynamic topics, and is used by the Processing Layer for the aforementioned data dissemination. It is a generic DDS service that facilitates the creation of dynamic DDS topics, following the OMG DDS Xtypes specification [OMG 2014]. It can be used by any application or service (not only D3CEP) that needs to perform DDS communication with Dynamic DDS Topics. With this service, a client can provide a dynamic data type description (i.e. a Metadata object instance, from the Metadata Sharing Service) to create a dynamic DDS topic correspondent to this type description, in execution type, without requiring the static description and compilation, at development type, of the DDS topic through an IDL. The OMD DDS XTypes [OMG 2014] is an extension to the OMG DDS [OMG 2006] specification which allows Topics to be defined dynamically, and at execution time, without requiring the compilation of an IDL file to generate classes and support types at development time. At the time of the writing of this work, the only DDS product that implements the XTypes specification is RTI Connext DDS [RTI 2016], from Real-Time Innovations, but is claimed by PrismTech that OpenSplice DDS [PrismTech 2016] will implement the specification soon. Since XTypes is a specification of the Object Management Group (OMG), and both these companies are the main players in DDS middleware, and are participant in the elaboration of the specification, it is expected that it will become a standard feature in all DDS implementations in the near future.

The D3CEP ***Processing Layer*** implements the event processing mechanisms, which supports the dynamic definition and deployment of EPAs, so as to compose distributed EPNs. Each Processing Layer instance contains a local CEP engine, which may have several deployed EPAs. The deployed EPAs process simple and complex

events received from outside the system and from other EPAs and generate derived events that are consumed by other EPAs (e.g. running locally or at other nodes) and by event consumers (i.e. subscribed monitoring applications or services). Each Processing Layer instance uses the Communication Layer for two purposes; the dissemination of D3CEP system entities that describe the configuration state of the system, such as EPNs, EPAs, deployment assignments of EPAs to processing nodes, among others; The dissemination of simple and complex events from event producers, consumers and among processing nodes. The EPAs deployed in a processing node use it to receive events as input, and to produce events as output into the DDS domain. The ***Deployment State Manager*** (DSM) manages the global deployment state of the D3CEP system. A DeploymentState is a representation of the global D3CEP system state, to which every processing node can access through the DDS global data space provided by the Communication Layer. The DSM on each processing node registers a listener into the D3CEP Entities DDS Service for receiving DeploymentState instances. When a DeploymentState is received from the network, the DSM gets the deployment information from it and verifies if there are EPAs assigned to be deployed to its local processing node (the id of the local node provided by the Processing Node Daemon). If there are EPAs to be deployed locally, the EPA reference is passed to the EPA Manager to be locally deployed. The ***EPA Manager*** manages the EPAs deployed locally in a processing node. A D3CEP_EPA has a set of rules, where each rule is composed of sequential D3CEP_Statements, which are Event Processing Language (EPL) statements describing the CEP rule processing logic. EPL was chosen since it is very similar to SQL, and is currently one of the most popular event processing languages. When the EPA Manager receives a D3CEP_EPA from the Deployment State Manager, it performs some steps to deploy the D3CEP_EPA into the local CEP Engine, and to plug the EPA input and output to the Communication Layer. After the Administrator has published a DeploymentState, and the Processing Layer in the processing nodes has performed all deployment steps, the EPAs deployed on each node are bound to Dynamic DDS Topics in the Communication Layer. The ***Event Types Manager*** performs the integration of event types at the Processing Layer with the Communication Layer, specifically for managing the mapping of event types at both levels. An Event Type instance is a representation of an event type, which contains the event type name, its Metadata description (obtained from the Metadata Sharing Service), and a correspondent Dynamic DDS Topic (obtained from the Communication Layer) for data dissemination through the DDS Domain. The D3CEP system is managed by an administrator, which is responsible for defining the description of the event types exchanged and processed by the system, CEP rules, EPAs, and EPNs, and the deployment of those entities on the distributed execution nodes of the processing core. In order to facilitate administration, two prototype applications have been developed: The Metadata Administrator and D3CEP Administrator. ***The Metadata Administrator Application*** facilitates the management of Metadata descriptions, by reading an XML file with the Metadata, and instantiating the Metadata Sharing Service and adding the Metadata to it, so it is published into the DDS domain. The ***D3CEP Administration Application*** facilitates the management of D3CEP Entities, by reading an XML file with those entities. Among the entities defined in the configuration repository are: The Deployment State to be set into the system, which contains EPNs, composed by EPAs. Each EPA has Statements, which are ordered EPL statements that implement the EPA's logic, external functions implemented in Java and called by these statements, a set of

ordered execution parameters, such as arguments to rules. Also, it contains the mapping of EPAs to processing nodes. References to external relational databases can also be defined. External databases can be used as knowledge bases for rules, or else, for storing events generated by the system, for example for auditing and generation of reports.

5. Related Work

The first initiatives in establishing a data-centric model for complex event processing are the ones addressing the integration of clustered DCEP systems using the DDS protocol in a static way, to which we refer to as a *Static Data-Centric* approach, where event types and rules are defined in a static way, and DDS topics are statically generated at compile time to support the data dissemination among data producers, consumers and processing nodes. In [Corsaro 2011c; Corsaro 2011a; Corsaro 2011b], the author presents the usage of OpenSplice DDS [PrismTech 2016] to integrate centralized Esper [EsperTech 2016] CEP engines to form a DCEP system (*OpenSplice+Esper*). Using a similar approach, in [RTI 2011], the RTI Connext DDS [RTI 2016] middleware is used to integrate Oracle CEP [Oracle 2016b] engines (*RTI+OracleCEP*), and in [Oberoi 2007] RTI Connext DDS is used to integrate Coral 8 CEP engines (*RTI+Coral8*). Following a static approach, no mechanism is defined for the definition and agreement of a common representation for event types, nor any dynamic mechanisms are available for the definition of those event types and CEP rules that use those event types. As the most direct relation to this thesis are DCEP systems that implement model similar to the one proposed to our approach, to which we call *Dynamic Integrated Model* approach, meaning that the processing and communication layers are strongly integrated/coupled. *Oracle CEP* is a DCEP system with a clustered deployment model. It is a system that implements capabilities that are similar to the ones provided by our approach. The system uses for data dissemination a distributed cache solution, called Oracle Coherence [Oracle 2016c], which provides a scalable, high-performance and fault-tolerant distributed cache. A distributed cache allows event instances to be directly referenced by any participant in the communication, so data producers, consumers and processing nodes read and write event instances to this distributed cache, which seamlessly propagates those instances to all participating nodes. In order to establish a common agreement for event representation, Oracle CEP provides an event types repository. Dynamic capabilities are achieved by the usage of the OSGI [Alliance 2016] component-based framework. The system provides graphical tools for defining event types and processing entities, such as EPNs, EPAs, etc. Another system with this approach is *Solar* [Chen et al. 2008], which is a middleware for pervasive and context-aware applications. Solar allows the system allows the construction of so-called Context Fusion Networks (CFN), which has a similar purpose as the EPNs in CEP. Data dissemination is established by a scalable and self-organizing peer-to-peer overlay network based on Pastry [Rowstron and Druschel 2001], which implements a distributed hash table, which propagates data streams from data producers, processing nodes and data consumers. The dynamically integrated approach of Solar is achieved by an attribute-based representation of data, which is agreed upon by all components in the system through the use of a distributed directory service, which provides naming and discovery functionalities. Therefore, dynamic capabilities in the definition of context processing queries are possible. We also have considered message-centric systems with dynamic capabilities, to which we call *Dynamic Message-centric* systems. TIBCO

Business Events [TIBCO Business Events 2016] is a DCEP system with a networked deployment, which uses other TIBCO products, such as the TIBCO Rendezvous [TIBCO 2016] publish/subscribe middleware for data dissemination, and allows the dynamic definition of events and rules, without stopping the system.

5.1. Comparison of Related Work

There are several systems and research efforts in the area of distributed complex event processing. Scalability, naturally, is the main motivation of all distributed solutions. Regarding the integration paradigm used, a few initiatives, such as the *OpenSplice+Esper*, *RTI+OracleCEP* and *RTI+Coral8*, show how to integrate DCEP using the DDS data-centric publish/subscribe middleware, but their approaches are static, which doesn't offer the flexibility required by the dynamic requirements of some monitoring applications. Other solutions, such as *Oracle CEP* and *Solar* follow a more evolved approach with dynamic aspects and also a global representation of event types to which all participants can agree to, so that applications can be developed and deployed dynamically using the systems. These dynamic integrated solutions use Distributed Caching and Distributed Hash Table respectively as their communication technologies for data dissemination. Among the message-centric solutions, not all work provide detail about the underlying technology being used, but among them are JMS implementations, point-to-point connections, and publish/subscribe implementations. Among the investigated message-centric solutions, only TIBCO Business events provide a fully integrated and dynamic model, although requiring the usage of proprietary TIBCO products. Table 1 shows a comparison table with the related work and the investigated aspects. The D3CEP solution, presented in this thesis, is included in the table, in order to facilitate comparison. A few of the investigated solutions provide capabilities for achieving a fully integrated and dynamic model for communication and processing of events.

Related Work	Integration Paradigm	Communication Technology	Dynamic
OpenSplice+Esper	Static Data-centric	OMG-DDS (OpenSplice DDS)	No
RTI+OracleCEP	Static Data-centric	OMG-DDS (RTI Connext)	No
RTI+Coral 8	Static Data-centric	OMG-DDS (RTI Connext)	No
Oracle CEP	**Dynamic Integrated**	**Dist. Cache (Oracle Coherence)**	**Yes**
Solar	**Dynamic Integrated**	**Dist. Hash Table (Pastry)**	**Yes**
D3CEP	**Dynamic Data-centric**	**OMG-DDS + XTypes (RTI Connext)**	**Yes**
TIBCO Business Events	**Dynamic Integrated**	**Prop. pub/sub (TIBCO Rendezvous)**	**Yes**

Table 1. Comparison of related work

These solutions are marked in bold in Table 1 (including the D3CEP system). Among these solutions, only TIBCO Business Events has a message-centric design, the others use a similar design approach, but using different protocols from our solution. The communication technologies used by these systems are differentiated from our approach, since we use the DDS protocol for implementing a dynamic and data-centric design. To the best of our knowledge, a fully integrated and dynamic approach for the definition and agreement of event types and processing entities with a data-centric paradigm, and using the DDS protocol, is not yet addressed by other work. By using the DDS protocol, we consider its advantages in performance, by relying on its optimized peer-to-peer routing of data, and capabilities in meeting requirements of a wide range of applications, considering its capabilities for supporting many different levels of performance and real-time strictness, which can be explored in industrial and mission critical scenarios. By providing a rich set of QoS policies, the DDS protocol can support future capabilities in defining QoS policies at detection level (i.e. for CEP rules), which

would reflect in the underlying communication layer, such as presented in [Appel et al. 2010], which describes a fully QoS aware infrastructure. Although this is not in the scope of this work, our solution provides a base framework that can be explored by future work in this aspect.

6. Use Case and Performance Evaluation

As an example use case, we define a scenario of a telemetry application, which receives data from mobile nodes (*MNs)*, such as vehicles. Those nodes periodically send a set of predefined raw events that can be used for the detection of many types of situations. For our scenario, we consider the events that contain geographical coordinates and the measurement error of these coordinates, to which we call *Location*. Therefore, with the system up and running, the Metadata Administrator application is run with an XML file defining this event type. For illustrating and evaluating performance and scalability, we then define a simple situation to be detected, but with a processing complexity that demands a distributed processing to share the load in order to handle an increasing number of MNs. Therefore, we assume a scenario of tracking MNs locations and triggering alerts when nodes are too close from one another. This situation could be used, for example, for air traffic control, situational awareness in command and control missions [T.A. DuBois et al. 2012; Baptista, Endler, et al. 2013] or for the detection of many types of mobility patterns [Baptista, Roriz, et al. 2013]. Thus, we define a complex event, called *TooClose*, which represent that a pair of MNs are within a specified distance D of each other. In order to create this event, we use the D3CEP Administration application to read an XML description defining an EPA called *TooClose, which* processes MNs Location events, combining all pairs of location events that are within distance D + 2ε from each other, where ε is the measurement accuracy. When this condition is true, the pair of Location events is aggregated into a TooClose complex event.

```
Too Close EPA
INSERT INTO TooClose
SELECT A, B
FROM Location.win:time_batch(30 sec) as A,
            Location.win:time_batch(30 sec) as B
WHERE A.nodeId <> B.nodeId AND
distance(A.latitude, A.longitude, B.latitude, B.longitude) < D + 2ε
```

Figure 3. CEP Rule of the Too Close EPA

Each EPA operates in a time-batch scheme, where it retrieves a stream window with all Location events in a given time period (e.g. all Location events from timestamp T to timestamp T - 30 seconds). The agent correlates this set of location events with the set itself, producing a product set of all possible pairs of location events. This result is then filtered by the distance function, that checks if each pair is close to each other, as shown in Figure 3. If the coordinates are within a minimum orthodromic distance (considering de accuracy of both coordinates), then the EPA produces an TooClose event containing the pair of *Location* events [Baptista, Roriz, et al. 2013]. Since the processing of the CEP rule in the ***TooClose EPA*** requires the product set of all possible pairs of Location events, in order to detect any node that is close to any other node, the considered area of interest in the map is divided into sub-regions that are considered by a single EPA, since it is only necessary to correlate the location of nodes in the same area. Therefore, we assume that each mobile node has a local function that is capable of using a location filter, defined in terms of latitude and longitude intervals, to assign an identification of a sub-region to each Location event. In this way, EPAs can only process *Location* events on the same sub-region, and many EPAs can be instantiated to support the detection of many sub-regions in a scalable manner.

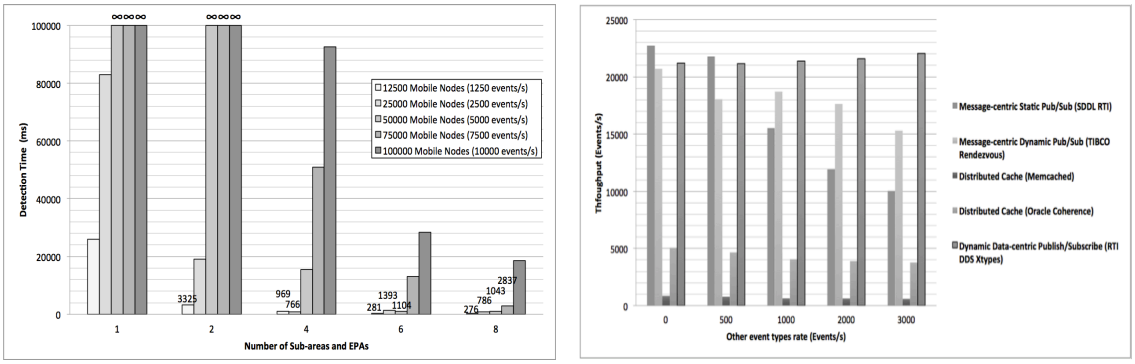


Figure 4. Detection time (a) and throughput (b) tests.

We distributed the processing among a set of processing nodes, showing how this can enable scalability of rules that have a high demand for processing. The simulation was executed in a delimited scope with sub-regions delimited by rectangles of a map, which represent the sub-region that each EPA subscribes. The application simulates 12500, 25000, 50000, 75000, 100000 MNs on this map. The number of nodes is divided among the number of instantiated EPAs (up to 8 EPAs). All those MNs send Location events that don't trigger the TooClose EPA rule, since they are on stationary locations that are distant from one another, and their goal is just to introduce load for the system to be tested. After all the nodes are sending data, a single pair of Location events that are actually close to one another is sent, with the goal of triggering the TooClose CEP rule. Since the MN Simulator is also a subscriber of TooClose events, when this event is triggered, the simulator measures the detection time of the rule. Figure 4a shows the results of the detection time test. The results reflect the scalability of the system, by increasing the number of processing nodes, the lower is the detection time. Instead of evaluating the performance of complete related DCEP systems, we have decided to isolate their communication layers, and use the same D3CEP processing layer, on top off those different communication technologies and integration models, so that the comparison is more direct. Different integration models DCEP systems that are able to support dynamic aspects were evaluated: The D3CEP system, with a **Dynamic Data-centric** model. With the ***Dynamic Message-Centric*** model: the TIBCO Rendezvous [TIBCO 2016] and Scalable Data Distribution Layer (SDDL) [David et al. 2012], which allow the dynamic definition of messages' structures. With the ***Dynamic Integrated*** model: we tested implementations of a Distributed Cache: Memcached [Fitzpatrick 2004] and Oracle Coherence [Oracle 2016c]. Therefore, we measure the throughput of the system using those different technologies, i.e. how many location instances per second are received by CEP engines. We established 3 EPAs that are capable of measuring the input rate of Location events. The MN Simulator is set to simulate 240000 mobile nodes sending events every 10 seconds, producing 24000 events per second. Three EPAs are set to receive and measure the input events rate. In addition to the Location events, the simulator produces an increasing rate of events of other types. So, we compare the sending of 0, 500, 1000, 2000 and 3000 events/s of types alternative to the Location event. Figure 4b shows the results for the throughput test. The results show the D3CEP system with the highest throughput, when the number of distinct event types increases. This is due to the fact that in a data-centric communication, in contrast to other models, the DCEP nodes don't need to open every message in order to inspect the contained event type. The data-centric middleware will only route the events with types that are in fact used by the deployed CEP rule.

7. Conclusion and Future Work

This paper presented a middleware for DCEP, with a data-centric and dynamic design approach based on the DDS specification, that supports on-line detection of patterns on cloud architectures. We have presented its architecture and its main aspects, and tests regarding performance and scalability. The use of DDS in our work provides base for future work using DDS for supporting DCEP, e.g. for IoT and mission critical applications, which may require QoS contracts at the detection level, as in [Appel et al. 2010].

8. References

Alliance, OSGI. 2016. OSGI. (2016). Retrieved April 10, 2016 from https://www.osgi.org/

Apache Software Foundation. 2016. Apache Kafka. (2016). Retrieved January 1, 2016 from http://kafka.apache.org/

Appel, Stefan, Sachs, Kai, and Buchmann, Alejandro. 2010. Quality of service in event-based systems. In *GvD Workshop*. Bad Helmstedt, Germany.

Baptista, Gustavo L.B., Endler, Markus, Dubois, Thomas A., and Johnson, Randall L. 2013. Middleware Supporting Net-Ready Applications for Enhancing Situational Awareness on Rotorcraft. In *9th International Conference on Autonomic and Autonomous Systems (ICAS 2013)*. Lisbon.

Baptista, Gustavo L.B., Roriz, Marcos, Vasconcelos, Rafael, Olivieri, Bruno, Vasconcelos, Igor, and Endler, Markus. 2013. On-line Detection of Collective Mobility Patterns through Distributed Complex Event Processing. *Monogr. em Ciência da Comput*. 13 (2013).

Byron, Ellis. 2014. *Real-Time Analytics: Techniques to Analyze and Visualize Streaming Data*, Wiley.

Chen, Guanling, Li, Ming, and Kotz, David. 2008. Data-centric middleware for context-aware pervasive computing. *Pervasive Mob. Comput*. 4, 2 (April 2008), 216–253.

Corsaro, Angelo. 2011a. Complex Event Processing with OpenSplice DDS. *PrismTech* (2011).

Corsaro, Angelo. 2011b. Getting Started with OpenSplice and Esper. *PrismTech* (2011).

Corsaro, Angelo. 2011c. Stream Processing with DDS and CEP. (2011). Retrieved April 10, 2016 from www.prismtech.com

Cugola, Gianpaolo and Margara, Alessandro. 2012. Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Comput. Surv*. (2012).

David, L., Vasconcelos, R., Alves, L., André, R., Baptista, G., and Endler, M. 2012. A Communication Middleware for Scalable Real-time Mobile Collaboration. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*. Toulouse, 54–59.

Erl, Thomas, Puttini, Ricardo, and Mahmood, Zaigham. 2013. *Cloud Computing: Concepts, Technology & Architecture*, Pearson Education.

EsperTech. 2016. Esper. (2016). Retrieved April 10, 2016 from http://esper.codehaus.org/

Etzion, O. and Niblett, P. 2010. *Event processing in action*, Manning Publications Co.

Fidler, E., Jacobsen, H.A., Li, G., and Mankovski, S. 2005. The PADRES Distributed Publish/Subscribe System. In *8th International Conference on Feature Interactions in Telecommunications and Software Systems*.

Fitzpatrick, Brad. 2004. Distributed caching with memcached. *Linux J*. (2004).

Greengard, Samuel. 2015. *The Internet of Things*, MIT Press.

Luckham, D.C. 2001. *The power of events: an introduction to complex event processing in distributed enterprise systems*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Nathan, Marz and Warren, James. 2015. *Big Data - Principles and best practices of scalable realtime data systems*,

OASIS. 2012. OASIS Advanced Message Queuing Protocol. , October (2012), 0–124.

Oberoi, Supreet. 2007. Introduction to Complex Event Processing & Data Streams. , August (2007), 20–23.

OMG. 2006. Data-Distribution Service for Real-Time Systems (DDS). (2006).

OMG. 2014. XTypes: Extensible and Dynamic Topic Types for DDS.November (2014).

Oracle. 2016a. Java Message Service. (2016). Retrieved April 10, 2016 from http://www.oracle.com/technetwork/java/jms/index.html

Oracle. 2016b. Oracle CEP. (2016). Retrieved April 10, 2016 from http://www.oracle.com/technetwork/middleware/complex-event-processing

Oracle. 2016c. Oracle Coherence. (2016). Retrieved April 10, 2016 from http://www.oracle.com/technetwork/middleware/coherence/overview/index.html

PrismTech. 2016. OpenSplice DDS. (2016). Retrieved April 10, 2016 from www.prismtech.com

Rowstron, Antony and Druschel, Peter. 2001. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Middlew. 2001* , November 2001 (2001), 329–350.

RTI. 2011. Integrating RTI Data Distribution Service Applications with Oracle CEP. *Real-Time Innov*. (2011), 1–9.

RTI. 2016. RTI Connext DDS. (2016). Retrieved April 10, 2016 from www.rti.com

Schneider, Stan. 2012. What ' s the Difference between Message Centric and Data Centric Middleware ? *Electron. Des*. (2012).

DuBois, Thomas, Blanton, Brendan, Reetz III, Ferdinand, Endler, Markus, Kinahan, William, Baptista, Gustavo, Johnson, Randall. 2012. Open Networking Technologies for the Integration of Net-Ready Applications on Rotorcraft. Annual Conference American Helicopter Society (2012).

Tambe, Sumant. 2012. Communication Patterns Using Data-Centric Publish-Subscribe. In *Silicon Valley Code Camp*.

TIBCO. 2016. TIBCO Rendezvous. (2016). Retrieved April 10, 2016 from http://www.tibco.com/products/automation/enterprise-messaging/rendezvous

TIBCO Business Events. 2016. TIBCO Business Events. (2016). Retrieved April 10, 2016 from http://www.tibco.com/products/event-processing/complex-event-processing/businessevents

Trilha Principal do SBRC 2016
Sessão Técnica 17
Redes Sem-fio

Atribuição dinâmica de canais em redes sem fio não coordenadas IEEE 802.11, baseada em fatores de sobreposição e intensidade de sinal

Alex Monteiro[1], Eduardo Souto[1], Richard Pazzi[2], Jussi Kiljander[3]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)

[2] Instituto de Tecnologia da Universidade de Ontario (UOIT) – Canadá

[3] Technical Research Centre of Finland (VTT) – Finlândia

{alex.monteiro,esouto}@icomp.ufam.edu.br,
richard.pazzi@uoit.ca,jussi.kiljander@vtt.fi}

Abstract. *IEEE 802.11 wireless networks have become one of the main responsible for the growing of mobile devices and their applications usage. This growth of devices, that use ISM band to traffic, created a competition for portions frequency available for use, resulting at the decreasing of traffic quality and increasing the amount of noise generated. This article describes a uncoordinated solution for IEEE 802.11 networks channel assignments based on adjacent channels overlap calculations combined with their signal strength. The tests showed that the solution had low convergence and throughput gains at the allocation of the best channels in scenarios that used spectrum at 2.4 GHz densely occupied.*

Resumo. *As redes sem fio IEEE 802.11 têm se tornado uma das principais responsáveis pelo crescimento do uso de dispositivos móveis e suas aplicações. Este crescimento de dispositivos, que utilizam a banda ISM para trafegar, gerou uma concorrência por porções de frequência disponíveis para sua utilização, acarretando na diminuição da qualidade de tráfego e no aumento da quantidade de ruído gerado. Este artigo descreve uma solução não coordenada para a atribuição de canais em redes IEEE 802.11, baseado em cálculos de sobreposição dos canais adjacentes combinada com sua intensidade de sinal. Os testes mostraram que a solução apresentou ganhos de vazão e baixa convergência na atribuição dos melhores canais em cenários com espectro em 2,4 GHz densamente ocupado.*

1 Introdução

A banda ISM 2,4 GHz (*Industrial, Scientific and Medical*) é uma das mais concorridas faixas de frequência não licenciada utilizadas atualmente. Compreendendo um intervalo de frequência que vai de 2,4 GHz até 2,5GHz, esta banda é largamente utilizada em redes sem fio que utilizam os padrões IEEE 802.11b/g/n. O padrão IEEE 802.11 divide a utilização da ISM 2,4 GHz em canais 14 canais (variando de acordo com a região), separando cada canal por 5 MHz e uma largura de banda de 22 MHz.

Esta divisão limitada de canais da banda ISM aliada ao crescimento da utilização de redes Wi-Fi como solução de conectividade em cenários domésticos e coorporativos fez com que problemas de reutilização de canal e interferências a partir de canais adjacentes tornassem-se muito mais frequentes. Este tipo de interferência geralmente é

causado pela má distribuição na escolha dos canais, o que impacta diretamente no desempenho da rede.

Esta má distribuição na escolha dos canais está diretamente relacionada ao processo de configuração dos APs (*Access Point*) ou WR (*Wireless Router*). A grande maioria dos APs disponíveis no mercado não é dotada de mecanismos inteligentes de atribuição de canal, cabendo ao usuário final ou ao administrador de rede a responsabilidade da escolha do canal a ser utilizado, sendo que em muitos casos estes usuários finais geralmente são leigos com relação à importância da correta e eficiente utilização do espectro.

Diferentes abordagens [Mishra *et al*, 2006] [Akl e Aurepally, 2007] [Gramacho *et al,* 2013] [Alquerque *et al*, 2013] têm sido propostas para tratar o problema da má distribuição de canais e propor uma utilização eficiente na banda ISM por redes Wi-Fi. Conforme [Mishra *et al*, 2006] e [Prya 2012] as principais abordagens adotadas são a centralizada (coordenada) e a distribuída (não coordenada). A abordagem centralizada assume um perfil mais corporativo, baseando seu funcionamento no uso de uma entidade centralizadora que controla e gerencia os canais utilizados pelo APs clientes. Na abordagem distribuída, cada AP seleciona e atribui o melhor canal a ser utilizado com base em informações dos APs vizinhos.

Apesar da abordagem centralizada ser interessante, por considerar as informações do ambiente de aplicação como um todo, falhas na entidade controladora podem comprometer (ou mesmo interromper) o processo de atribuição de canais. Além disso, na prática, as redes sem fio são extremamente heterogêneas e geralmente administradas por diferentes usuários. Tais características tornam as soluções baseadas em abordagens distribuídas mais interessantes e apropriadas para os cenários onde não existe uma entidade controladora central como os comumente encontrados nos ambientes residenciais. Neste contexto, este trabalho propõe um novo mecanismo de seleção e atribuição de canais, onde cada nó (AP) na rede executa sua própria cópia do algoritmo baseado em informações coletas das redes vizinhas para atribuir o seu canal.

A principal contribuição deste trabalho está no projeto e na avaliação de uma solução de atribuição dinâmica de canais em redes IEEE 802.11 não coordenadas, onde cada rede não possui nenhum vínculo de comunicação com a outra, operando de forma independente. Esta solução trabalha de forma descentralizada, adequando-se à dinamicidade da utilização do espectro em tempo real, de forma estável e transparente, utilizando *hardwares* comuns encontrados do mercado.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados às técnicas de atribuição de canais para redes IEEE 802.11. A Seção 3 apresenta a métrica de avaliação e o algoritmo para atribuição do melhor canal utilizado neste trabalho. A Seção 4 descreve os experimentos e os resultados obtidos. Por fim, a Seção 5 apresenta as conclusões deste trabalho.

2 Trabalhos Relacionados

De modo geral, os trabalhos encontrados na literatura que propõem soluções para atribuição dinâmica de canais se baseiam em duas principais questões; quais métricas utilizar para selecionar o melhor canal e quem decidirá sobre a mudança de canal (decisão centralizada ou distribuída)? A seguir serão apresentados alguns dos principais

trabalhos que propõem soluções de alocação distribuídas, organizados de acordo com as métricas utilizadas para processo de seleção e atribuição de canal.

[Gramacho *et al.*, 2013] definem como melhor canal aquele que é capaz de propiciar a maior velocidade de comunicação. Para tal, os autores propõem um modelo de interferência que considera as contenções de transmissão e degradação da recepção dos dados. Assim, o modelo leva em consideração as diferenças no perfil de interferência de clientes em relação ao AP. No entanto, nesta abordagem torna-se necessária a utilização de um componente auxiliador (NMON) para solicitar medições de varredura, executar o algoritmo de seleção de canal e comandar a troca de canal no AP associado.

[Boumalf *et al.*, 2008] propõem um algoritmo de alocação de canais baseado na métrica SIR (*Signal to Interferences Ratio*). O objetivo dos autores é minimizar o total de interferência entre os APs mantendo um controle sobre a taxa de interferência de sinal. O algoritmo inicia a distribuição do canal para os APs calculando o SIR para cada usuário utilizando algoritmo PLI (Programação Linear Inteira) e o coeficiente de sobreposição de canal em relação aos APs vizinhos. O algoritmo proposto apresenta como resultado uma melhoria significativa da SIR sobre WLANs.

[Akl e Arepally, 2007] selecionam o melhor canal com base em um modelo de interferência que relaciona fator de sobreposição de interferência de canal, potência de transmissão e a perda de potência pela distância dos APs, como o objetivo de minimizar a interferência entre eles e maximizar o desempenho da rede. Uma característica interessante deste algoritmo é que ele pode ter dois tratamentos diferentes quando houver um conjunto de melhores canais. No primeiro caso, o algoritmo escolhe o primeiro canal do conjunto e no segundo caso seleciona um dos canais randomicamente.

[Zhou e Liu, 2012] propõem um algoritmo semelhante ao apresentado por [Akl e Arepally, 2007], utilizando como métrica apenas o fator de sobreposição. O objetivo do algoritmo é maximizar o menor SIR, através da redução da interferência co-canal também utilizando Programação Linear Inteira.

[Silva et al., 2006] propõem um mecanismo de alocação de canais que leva em consideração a interferência de redes vizinhas. A quantificação da interferência considera a utilização do canal e o nível de ruído. O algoritmo de atribuição de canais é baseado em informações coletadas pelos clientes de redes IEEE 802.11k.

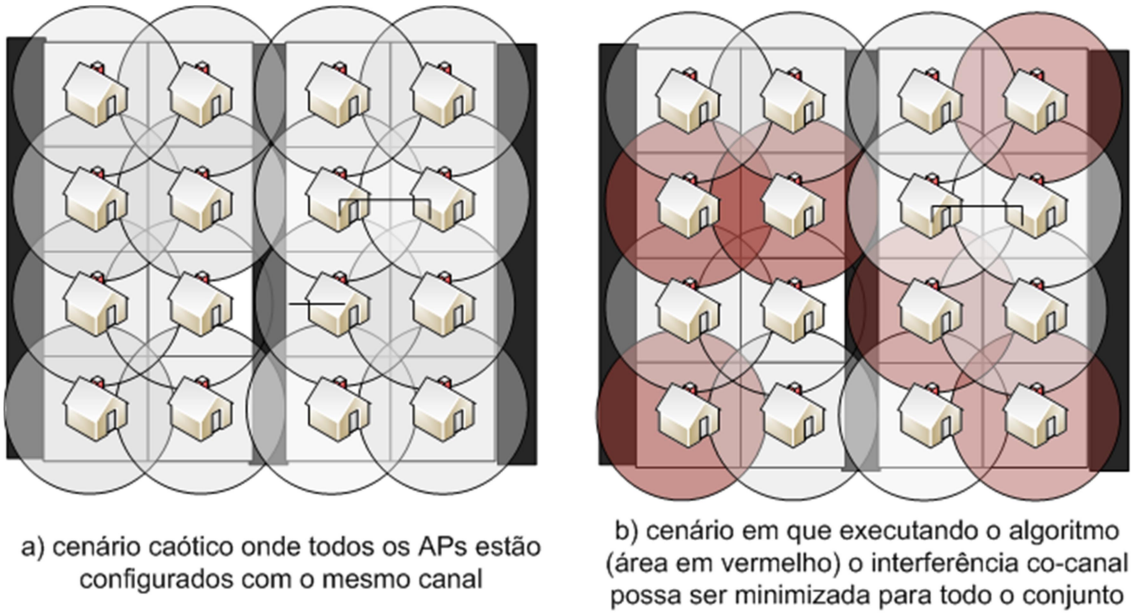
O presente trabalho se destaca entre as demais, uma vez que no mesmo foi utilizado um testbed em cenário real de teste utilizando roteadores comuns disponíveis no mercado com firmware OpenWRT. Além disso, utiliza-se uma abordagem não coordenada, considerando cada AP como uma rede independente, evitando a necessidade de uma nó controlador, para sua seleção e atribuição de canal, permitindo que o mecanismo possa ser executado tanto em redes residenciais como em redes corporativas.

3 Solução para atribuição dinâmica de canais em redes 802.11 não coordenadas

Esta seção descreve um novo mecanismo de atribuição de canais em redes 802.11 baseado em um modelo de interferência FSI (Fator de Sobreposição e Intensidade) que pontua cada canal com base no seu peso de utilização. Este modelo combina o fator ou

coeficiente de sobreposição de canal proposto em [Burton 2002] e também utilizado em [Akl e Aurepally, 2008], [Bolfmalf *et al.*, 2007] e [Zhou e Liu, 2012], com o fator de intensidade de sinal. O peso de utilização é obtido a partir do somatório do produto do fator de sobreposição de canal pela intensidade de sinal de cada AP. Desta forma, o canal com menor peso corresponderá ao canal que possui a menor interferência co-canal e melhor intensidade de sinal, podendo ser escolhido como o melhor canal a ser utilizado no instante de tempo *t*. Com a atribuição dos canais com o menor peso de utilização por parte dos APs participantes da solução, a interferência co-canal será ainda mais reduzida fazendo com que ocorra um ganho de vazão nas redes sem fio participantes ou não da solução.

A Figura 1 (a) descreve o cenário caótico tratado pelo mecanismo proposto. A adesão de redes sem fio em ambientes residenciais, em geral, ocorre de forma desordenada e dinâmica, levando no pior caso a configuração de todas as redes sem fio no mesmo canal. O mecanismo proposto visa atribuir os melhores canais para os APs que adotam a solução e, como consequência, minimizar o congestionamento dos canais utilizados por redes que não possuem o mecanismo, conforme ilustra a Figura 1 (b).



a) cenário caótico onde todos os APs estão configurados com o mesmo canal

b) cenário em que executando o algoritmo (área em vermelho) o interferência co-canal possa ser minimizada para todo o conjunto

Figura 1. Cenário hipotético tratado pelo mecanismo proposto.

Para lidar com o cenário apresentado, o mecanismo de atribuição de canais proposto leva em consideração todos os canais disponíveis (sobrepostos e não sobrepostos), sendo flexível com a região de utilização, ou seja, pode ser ajustado para o padrão Americano (11 canais), Europeu (13 canais) e Japonês (14 canais).

As subseções seguintes detalharão o modelo de interferência utilizado para a avaliação do(s) melhor(es) canal(is) e o algoritmo desenvolvido para seleção e atribuição do melhor canal.

3.1 Modelo de interferência FSI (Fator de Sobreposição e Intensidade)

Quando se deseja analisar um determinado sinal, três importantes parâmetros são utilizados como base para esta análise, são eles: a frequência, o nível do sinal e largura de banda do canal. A Figura 2 apresenta os canais para o padrão IEEE 802.11 na banda

de uso não licenciada (ISM) na frequência de 2,4 GHz. Este padrão divide a banda ISM em 14 canais, onde sua utilização pode variar de acordo com a região, com uma largura de banda fixa de 22 MHz.

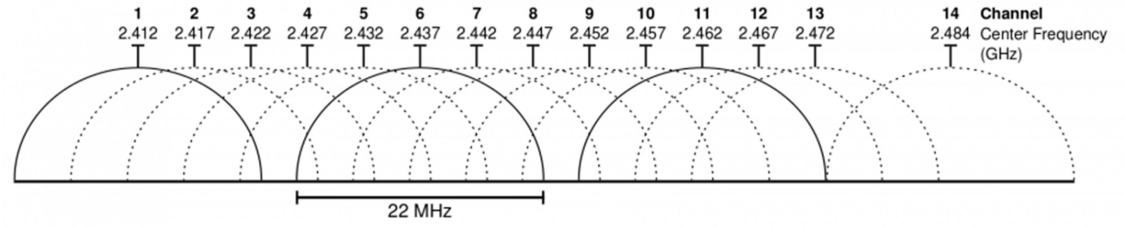


Figura 2. Canais IEEE 802.11 na banda ISM 2.4 GHz [Chieochan *et al.*, 2010].

É importante salientar que outros padrões de comunicação como os padrões 802.15.1 e 802.15.4 também utilizam esta faixa de frequência, bem como outros dispositivos como os telefones sem fio e controles remotos. Portanto, a coexistência de tecnologias baseadas nesses padrões pode causar interferência nas redes sem fio 802.11. Contudo, este tipo de interferência não será levado em consideração no cálculo do fator de sobreposição (f_S) utilizado neste trabalho.

O fator de sobreposição $f_{S(i,j)}$ é o resultado da porcentagem relativa de interferência de dois APs i e j utilizando canais sobrepostos. Os canais sobrepostos atribuídos para os APs devem ser cuidadosamente escolhidos.

O fator de sobreposição de canal é definido como:

$$f_S(i,j) = \begin{cases} 1 - \frac{|F_i - F_j|}{w} & se\ f_S(i,j) \geq 0 \\ 0 & senão \end{cases} \qquad \text{(Equação 1)}$$

onde F_i é a frequência do canal atribuída ao AP i e F_j é a frequência do canal atribuída ao AP j, w é largura do canal. Por exemplo, se o canal 1 é atribuído ao AP i e o canal 1 também é atribuído AP j, o fator de sobreposição $f_S(i,j)$, desses APs é 1.0, ou seja 100%. Por outro lado, se o canal 5 fosse atribuído ao AP j, $f_S(i,j)$ seria 0,09, ou seja, 9%. Caso o canal 6 ou canais superiores fossem atribuídos ao AP j, $f_S(i,j)$ seria de 0.0, significando que não existe interferência entre o AP i *e* AP j, para este cenário.

Tabela 1 - Matriz de sobreposição de canais

Matriz de sobreposição														
Canal	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	1	0.77	0.54	0.31	0.09	0	0	0	0	0	0	0	0	0
2	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0	0	0	0	0
3	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0	0	0	0
4	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0	0	0
5	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0	0
6	0	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0
7	0	0	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0
8	0	0	0	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0
9	0	0	0	0	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0
10	0	0	0	0	0	0.09	0.31	0.54	0.77	1	0.75	0.5	0.31	0
11	0	0	0	0	0	0	0.09	0.31	0.54	0.77	1	0.75	0.54	0
12	0	0	0	0	0	0	0	0.09	0.31	0.54	0.77	1	0.77	0.22
13	0	0	0	0	0	0	0	0	0.09	0.31	0.54	0.77	1	0.45
14	0	0	0	0	0	0	0	0	0	0	0	0.22	0.45	1

Realizando o mesmo cálculo para cada canal em relação aos demais é possível construir uma matriz de sobreposição de canais, que representa o fator de sobreposição de cada canal em relação aos demais, conforme a Tabela 1. Perceba que a relação de sobreposição de um determinado canal em relação a ele mesmo é igual 1, ou seja, caso duas redes utilizem o mesmo canal a sobreposição de canal será de 100% em relação a frequência.

O nível de intensidade de sinal I_R é calculado pela subtração da intensidade do sinal pelo nível de menor sensibilidade de recepção. Este trabalho define o menor nível de sensibilidade como -100 dBm. A maioria das interfaces Wi-Fi existentes no mercado adota como valor limite para sensibilidade de recepção entre -90 a -100 dBm. Desta forma, se um determinado canal possui um nível de sinal de -35 dBm sua intensidade de sinal será obtida pela Equação 2:

$$I_R = -35 - (-100)\ \text{dBm} = 75 \qquad \text{(Equação 2)}$$

O peso de utilização P_u é um somatório do produto da relação de sobreposição de canal, que corresponde a divisão da faixa de interseção entre os canais e a largura de banda, e o somatório da intensidade dos sinais de determinado canal, conforme a Equação (3):

$$P_U(canal) = \sum_{i=0}^{14} \left(M_{fs}(canal, i) * (\sum_{red}^{n} \quad I_R(rede, i)) \right) \text{(Equação 3)}$$

Por exemplo, considerando que um determinado AP detectou três redes distintas nos canais 1, 2 e 4, com os níveis de sinais -40 dBm, -50 dBm e -35 dBm, respectivamente. O cálculo do peso de utilização para cada canal será:

Para o canal 1:

$$P_U(1) = M_{fs}(1,1) * \big(-40 - (-100)\big) + M_{fs}(1,2) * \big(-50 - (-100)\big) + M_{fs}(1,2) * \big(-50 - (-100)\big)*$$

$$P_U(1) = 1 * 60 + 0{,}77 * 50 + 0{,}31 * 65 = 118{,}65$$

Para o canal 2:

$$P_U(2) = M_{fs}(2,1) * \big(-40 - (-100)\big) + M_{fs}(2,2) * \big(-50 - (-100)\big) + M_{fs}(2,4) * \big(-50 - (-100)\big)*$$

$$P_U(2) = 0{,}77 * 60 + 1 * 50 + 0{,}54 * 65 = 131{,}3$$

Para o canal 4:

$$P_U(4) = M_{fs}(4,1) * \big(-40 - (-100)\big) + M_{fs}(4,2) * \big(-50 - (-100)\big) + M_{fs}(4,4) * \big(-35 - (-100)\big)$$

$$P_U(4) = 0{,}31 * 60 + 0{,}54 * 50 + 1 * 65 = 110{,}00$$

A Tabela 2 apresenta o cálculo para os demais canais.

Tabela 2. Pesos de utilização por canal

Canal	1	2	3	4	5	6	7	8	9	10	11	12	13
Fator	118,65	131,3	120,95	110	70,95	39,6	15,5	5,85	0	0	0	0	0

Canais disponíveis
Canais em utilização

Caso houvesse a necessidade da inserção de um novo AP na rede e considerando os pesos de utilização de canais dispostos na Tabela 2, os melhores canais (canais candidatos) para atribuir a este novo AP seriam os canais 9, 10, 11, 12 e 13.

3.2 Algoritmo de atribuição de canal autônomo

O Quadro 1 apresenta um pseudocódigo para o algoritmo de atribuição de melhor canal proposto, aqui denominado de DCA (*Dynamic Channel Assignment*). Este algoritmo é executado a cada nova varredura do espectro, realizando o cálculo dos fatores de sobreposição e a atribuição do melhor canal do AP. Uma função *varredura* é definida para analisar o espectro em busca de redes sem fio, preenchendo o resultado dessa análise em uma estrutura de dados denominada "listaderedes" (linha 7). Dois vetores são utilizados para o cálculo dos fatores de sobreposição, o vetor "utilizacao" e o vetor "fatoresdesobreposicao" (linhas 2 e 3) cada um com o tamanho igual à quantidade de canais possíveis. A função *utilizacao()* recebe a lista de redes e atualiza o vetor utilização com o somatório das intensidades das redes detectadas de cada canal (linha 8).

Em seguida, a função "*calculafatordesobreposicao()"* (linha 9) recebe como entrada o vetor de utilização e executa o cálculo do fator de sobreposição para cada canal, atribuindo o resultado no vetor "fatoresdesobreposição. Com o cálculo do fator de sobreposição realizado, armazena-se o valor do canal atual na variável "canal_atual" (linha 10) e na estrutura "melhores_canais" (linha 11) os melhores canais, que pela função melhores_canais() são o canais com menor fator de sobreposição.

A fim de garantir estabilidade ao sistema, o algoritmo DCA possui uma estrutura de controle que verifica a constância da escolha do melhor canal, evitando que haja uma troca de canal a cada nova leitura em ambientes extremamente dinâmicos. Nesta estrutura verifica-se o canal atual já está contido na lista de melhores canais (linha 13), se já estiver define a variável "histerese" igual a 0 (zero) (linha 17) aguarda um determinado tempo e executa o algoritmo novamente. Se o canal atual não estiver na lista de melhores canais ele verifica se o melhor canal escolhido é equivalente ao melhor canal escolhido anteriormente (linha 14), caso verdadeiro incrementa a variável "histerese" (linha 15), caso falso atribui zero a esta variável (linha 17). A efetiva atribuição de canal, ou seja, a troca do canal será executada somente se partir da *h*-ésima amostra o valor melhor canal permanecer constante.

4 Experimento e Resultados

O algoritmo de atribuição de canais desenvolvido foi projetado para utilização em hardwares comuns disponíveis no mercado que permitam a utilização do firmware OpenWRT. A solução foi validada em uma rede IEEE 802.11g no modo infraestrutura no laboratório de Segurança de Sistemas e Tecnologias Emergentes da Universidade Federal do Amazonas. Para o experimento, foram utilizados 4 (quatro) roteadores Linksys WRT54GL com sistema operacional embarcado OpenWRT 10.03 *Backfire*, no modo AP (Ponto de Acesso) abordando dois cenários indoor diferentes, o cenário 1 quando apenas 1 roteador utiliza o mecanismo de alocação dinâmica e no cenário 2 quando os quatro roteadores executam o mecanismo simultaneamente. Além dos roteadores outras 4 *workstations* foram utilizadas como clientes da rede, cada uma delas associada a um respectivo roteador.

```
1 estrutura listaderedes;
2 vetor real matrizdesobreposicao[13][13];
3 vetor inteiro utilizacao[13], melhores_canais;
4 vetor real fatoresdesobreposicao[quantidade de canais];
5 inteiro canal_atual, histerese, anterior_mc;
6       para infinito{
7             varredura(listaderedes);
8             utilizacao(listaderedes);
9             fatoresdesobreposicao = calculafatordesobreposicao(utilizacao,matrizdesobreposicao);
10            canal_atual = ler_canal_atual();
11            melhores_canais = melhor_canal(fatorresdesobreposicao);
12
13           se (canal_atual != melhores_canais){
14                  se (melhores_canais = anterior_mc)
15                        histerese = histerese + 1;
16                  senão{
17                        histerese = 0;
18                        anterior_mc = melhor_canais
19                  }
20                  se ( histerese == h)
21                        alterar_canal(melhor_canais);
22                  senão
23                        histerese = 0;
24            }
25
26      aguarda(tempo);   // Tempo = intervalo de análise
27
28      }fimdopara
```

Quadro 1 – Pseudocódigo do Algoritmo de Alocação Dinâmica de Canais (DCA) baseada no fator de sobreposição e intensidade.

O objetivo principal desta avaliação é verificar o desempenho em relação aos canais estáticos e a convergência na escolha dos canais em relação aos outros roteadores utilizando o mecanismo simultaneamente, tendo em vista que a abordagem do mecanismo proposto trata o problema de escolha do melhor canal de forma não coordenada. As vantagens e ganhos deste mecanismo é demonstrado através da comparação da vazão de dados dos canais estáticos em relação ao algoritmo utilizado neste trabalho, e também da comparação entre a melhor distribuição de canais que pode ser utilizada em relação a adotada pelo conjunto de escolha dos roteadores.

A Figura 3 apresenta o *layout* do laboratório ETSS (*Emergent Technologies and Security Systems*), na qual a rede de testes foi implantada. Nesta planta os nós clientes participantes são representados por estações de trabalho na cor preta, o vínculo de comunicação sem fio por um raio em amarelo e o ponto acesso em azul. A ocupação do espectro próximo ao redor do laboratório sem a inserção do APs do experimento está distribuída conforme a Tabela 3. Estes dados foram coletados pelo comando “iwlist” em uma *workstation* com distribuição Ubuntu 12.04.

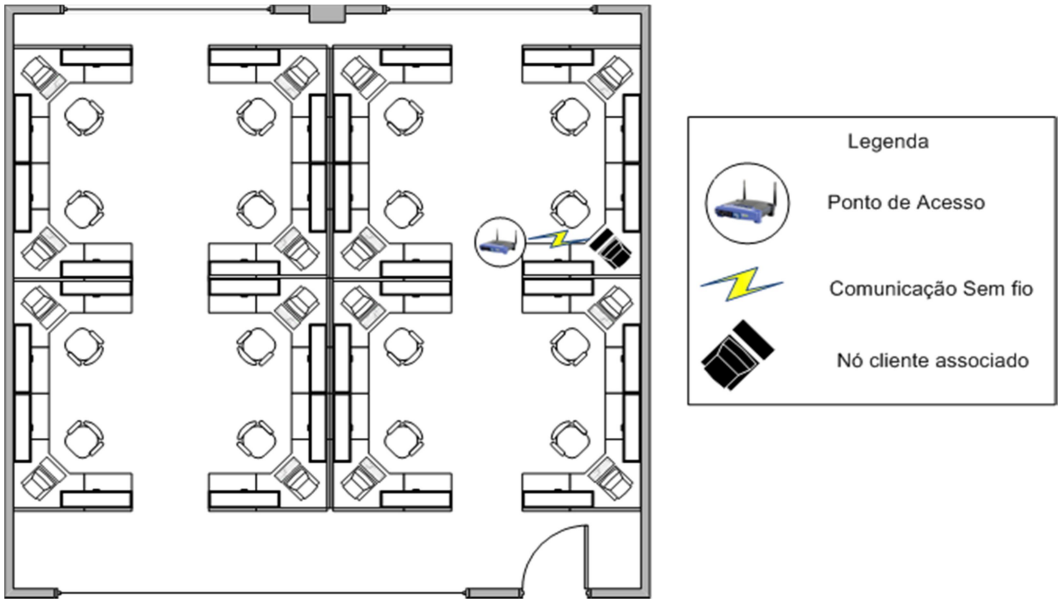


Figura 3 – Cenário de testes 1, composto por uma *workstation* e um roteador Linksys WRT54GL com o algoritmo DCA embarcado.

O teste de vazão sem o algoritmo proposto para cada canal foi gerado e mensurado com a utilização do programa *Iperf* [Iperf], para gerar tráfego UDP partindo do ponto de acesso para seu cliente associado, foram coletadas um total de 600 amostras em um período de 10 minutos. A Figura 4 apresenta a vazão por minuto dos canais de 1 ao 13, na gráfico é possível que observar que os canais ortogonais 1 e 11 apresentam a melhores vazões médias por minuto. Esta informação é importante pois será utilizada como *baseline* para comparação com o algoritmo DCA.

Tabela 3 - Ocupação do espectro na faixa 2,4GHz, por motivos de segurança os SSID reais das redes sem fio participantes do experimento foram substituídas por letras.

ESSID	Canal	Frequência	Qualidade	Sinal (dBm)	Max. Bitrate Mbps	Modo
Rede A	13	2.472 GHz	83%	-52	54	Infraestrutura
Rede B	1	2.412 GHz	83%	-52	54	Infraestrutura
Rede C	1	2.412 GHz	81%	-53	54	Infraestrutura
Rede D	1	2.412 GHz	80%	-54	54	Infraestrutura
Rede E	4	2.427 GHz	81%	-53	54	Infraestrutura
Rede F	11	2.462 GHz	83%	-52	54	Infraestrutura
Rede G	11	2.462 GHz	83%	-52	54	Infraestrutura
Rede H	9	2.452 GHz	81%	-53	54	Infraestrutura
Rede I	10	2.457 GHz	77%	-56	11	Ad-hoc
Rede J	10	2.457 GHz	77%	-56	54	Infraestrutura
Rede L	6	2.437 GHz	83%	-52	54	Ad-hoc
Rede M	6	2.437 GHz	81%	-53	54	Infraestrutura
Rede N	6	2.437 GHz	81%	-53	54	Infraestrutura
Rede O	6	2.437 GHz	79%	-55	54	Infraestrutura

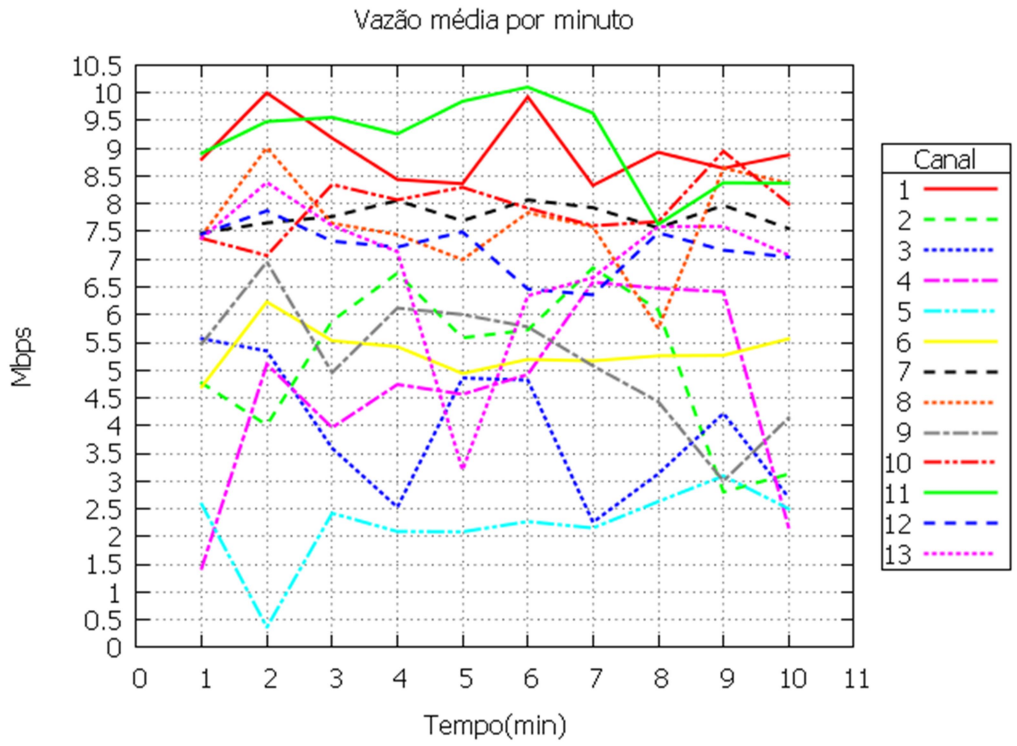


Figura 4 – Vazão média por minuto de cada canal.

Para o algoritmo DCA, os mesmos parâmetros experimentais da ferramenta Iperf foram utilizados. Para o experimento, foi definido um intervalo de tempo de 15 segundos entre cada análise de espectro realizada pelo algoritmo, e um valor de histerese *h* igual a 3, de modo que fossem realizadas 4 análises de espectro por minuto, permitindo uma troca de canal por minuto, quando necessário. Para o experimento do algoritmo proposto definimos o canal 6 como canal inicial, esta escolha foi motivada porque geralmente este canal é configurado de fábrica por diversos fabricantes de roteadores e pontos de acesso. Para o cenário 1, o algoritmo DCA proposto foi testado em 5 iterações.

Os resultados, apresentados na Figura 5, mostraram um ganho de 31% em relação ao canal inicial, mas inferior aos canais 1 e 11. Apesar da escolha do melhor canal convergir para o canal com as melhores vazões, o algoritmo não alcançou valores próximos a vazões destes canais. Verificando o motivo desta discrepância, constatou-se que o hardware utilizado no experimento pode influenciar significativamente nos resultados, o roteador WRT54GL possui apenas uma única interface de rádio, quando o algoritmo realiza uma varredura do espectro, ele gera uma interrupção na interface de rede, parando o tráfego de dados neste período de tempo, afetando assim o desempenho da transmissão de dados. Acreditamos que nossos resultados poderão ser superiores quando testado em hardware mais robusto, que comporte duas interfaces de rede, assim uma das interfaces poderia ser utilizada exclusivamente para sensoriar o espectro, permitindo reduzir o intervalo de análise e seleção do melhor canal, e maior flexibilidade em ambientes mais dinâmicos; ou aumentando o tempo do intervalo de varredura.

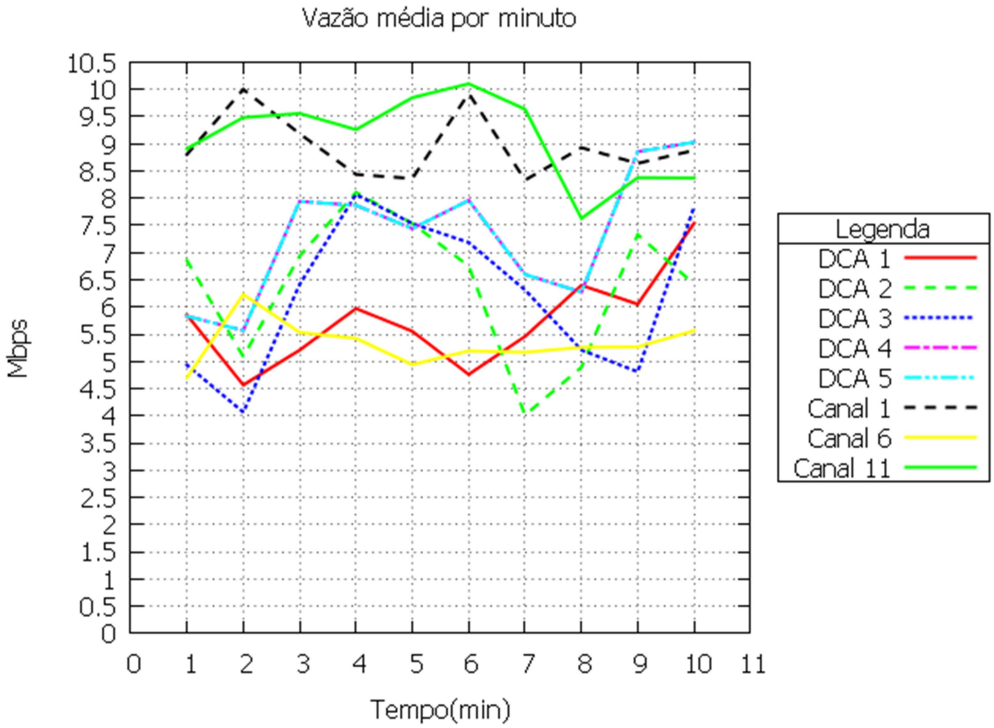


Figura 5 - Vazão média por minuto do algoritmo DCA em comparação aos canais ortogonais 1, 6 e 11 no Cenário 1.

No cenário 2, Figura 6, foram utilizados 4 roteadores WRT54GL, neste cenário o AP 1 foi definido com o canal 1, os APs 2, 3 e 4 no canal 6, foram seguidos os mesmos padrões de testes do cenário 1 com o programa *Iperf*, observando agora um conjunto de APs. O objetivo agora era estressar o algoritmo, a fim de verificar a vazão do conjunto e a convergência na escolha dos canais, que em outras palavras significa verificar se todos os APs convergiram para utilização um mesmo canal (o que seria ruim), ou se eles optaram por uma distribuição que seja adequada ao grupo de APs.

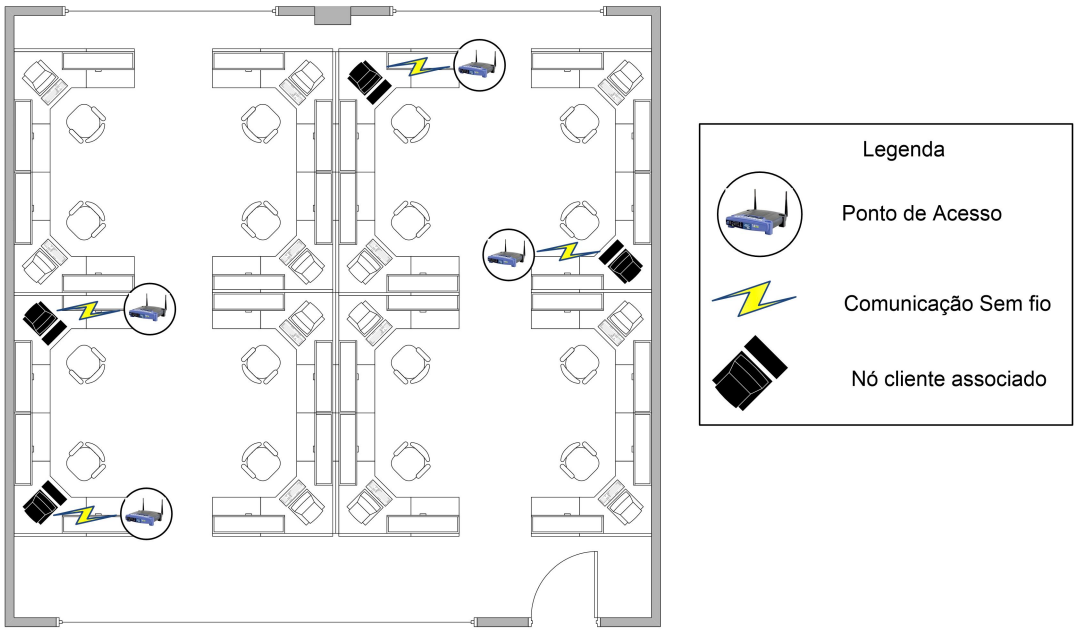


Figura 6 - Cenário 2

Nossos experimentos mostraram ganhos de vazão posteriores a todas as trocas de canais. É possível observar, conforme a Figura 7, no AP01 como não houve a necessidade de troca de canal, por já estar no melhor canal sua vazão permaneceu estável na maior parte do tempo. No AP02 e AP03 a cada troca realizada suas vazões aumentaram consideravelmente. No AP04 devido as trocas de canais pelo outros APs, este AP permaneceu até o próximo ao instante 500 no canal 6, o canal densamente ocupado, mas começou apresentar ganhos a partir da troca pelo canal 10.

A Figura 8 apresenta a seleção do melhor canal (em vermelho) pelo canal atual do AP (linha de cor preta). Foi observado que o comportamento do AP1 foi o mais estável realizando apenas uma troca, alterando do canal 1 para o 13. O APs 2, 3 e 4 começaram a sofrer a influência da escolha do AP1 a partir da amostra 100 quando a alteração de canal realizada por este altera a ocupação do espectro nas frequências próximas do canal 13. No intervalo entre as amostras 100 e 450 é possível observar que em relação aos canais iniciais ocorreu apenas um rearranjo na distribuição dos canais previamente configurados. A partir da amostra 500 o processo de seleção de canal começa a se estabilizar definindo em 75% dos roteadores com o canal 13 e 25% com o canal 1.

A convergência de canais é uma das desvantagens do processo de seleção de canal não coordenado. Como as escolhas de cada AP ocorre de forma individual, é importante observar que quando todos são iniciados simultaneamente suas escolhas podem convergir para um único canal, caso todos estejam configurados com o mesmo canal inicial. Nossos resultados mostraram que nosso algoritmo apesar de não negociar os canais com os outros pontos de acesso participantes da solução, encontrou uma distribuição em que não houvesse convergência de canais, conforme ilustra as Figuras 7 e 8, os APs 1,2,3,4 respectivamente nos canais 1,6,6,6 concluíram os experimentos nos canais 1,13,1,10. Evitando assim a utilização de canais com alta interferência.

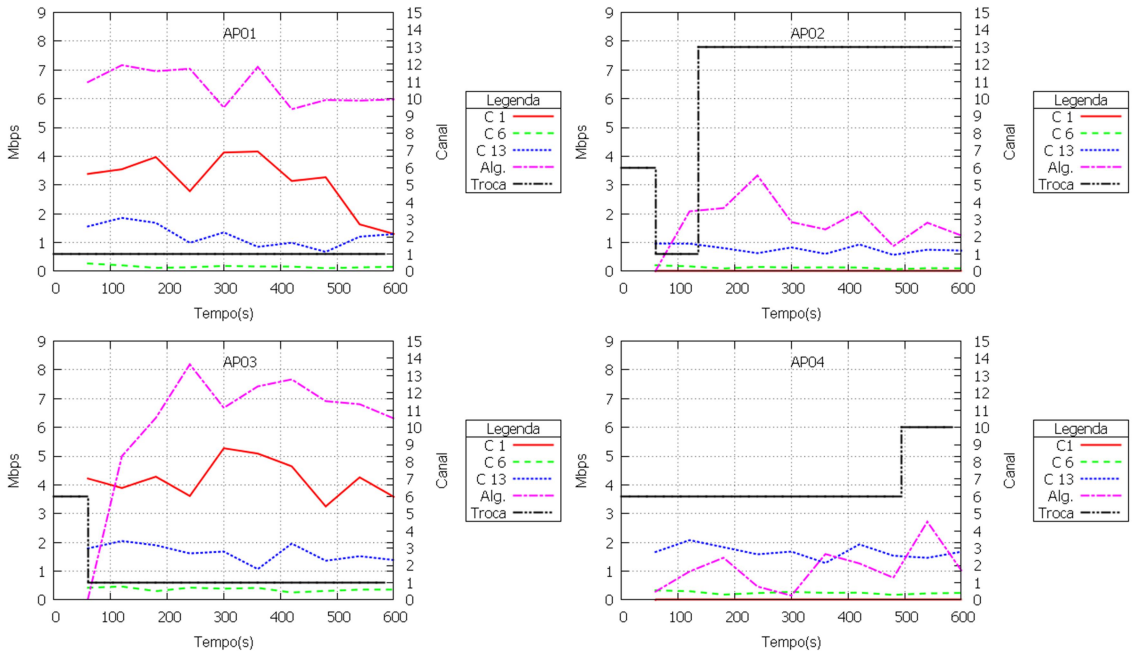


Figura 7 - Vazão do algoritmo por ponto de acesso

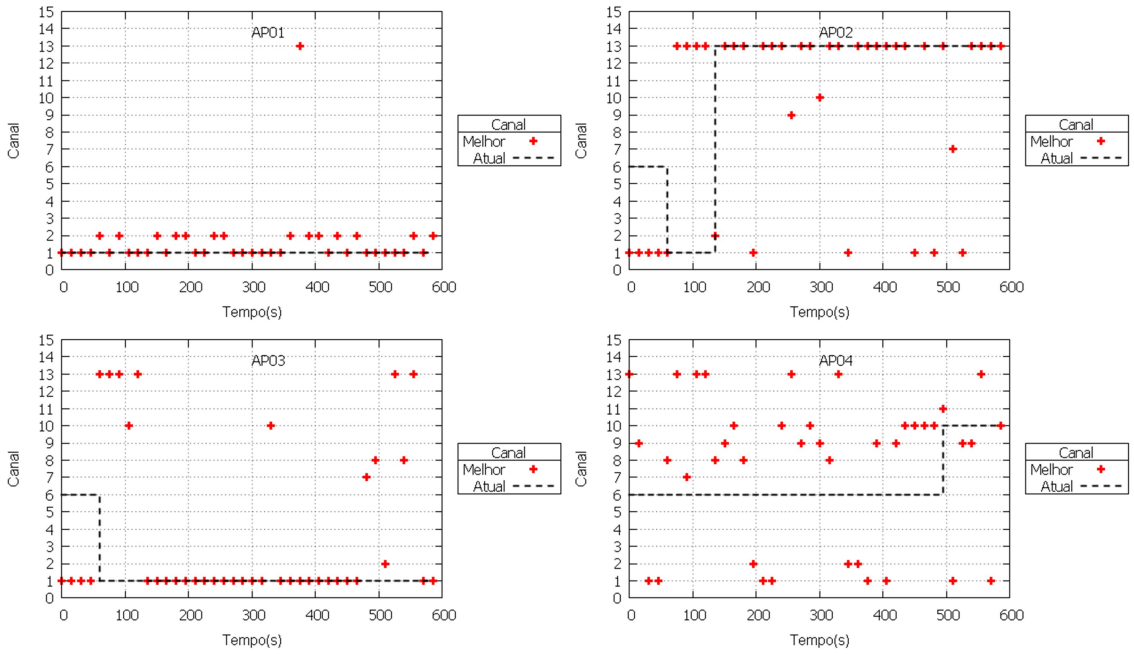


Figura 8 - Canal atual por troca efetiva

As taxas de permanência no melhor canal (relação entre o resultado do melhor canal do algoritmo pelo canal atual do AP) para o APs 1, 2, 3 e 4 foram de 92%, 81%, 59% e 78,5%, respectivamente. A solução proposta apresentou uma taxa média de 77% de permanência entre o canal selecionado e o canal atual dos APs.

5 Considerações finais

Soluções de atribuição dinâmica de canais utilizando uma abordagem centralizada possui uma vantagem de destaque em relação à descentralizada. Elas permitem que nó controlador possa ter um panorama geral da disposição dos APs, de como eles se relacionam e como está a distribuição do espectro. Em contrapartida, cenários compostos por redes não coordenadas (residenciais) são muito mais frequentes, fazendo com que a abordagem centralizada não seja a mais adequada para este tipo de cenário, pois em um cenário assim, nenhuma rede possui um vínculo de comunicação com a outra, não havendo possibilidade de existir um nó controlador, a menos que seja na nuvem.

Este artigo apresentou um algoritmo e um modelo de interferência, denominado FSI, que permite atribuir o melhor canal a cada AP sem necessidade de negociação na escolha de canais, de modo a diminuir a interferência co-canal, resultando no aumento de vazão de todos APs participantes da solução ou não. O algoritmo foi implementado e testado em um testbed real, utilizando roteadores em modo ponto de acesso que permitem utilizar o *firmware OpenWRT*.

Os experimentos mostraram ganhos médios superiores a 30% em relação aos canais anteriormente configurados, mesmo utilizando *hardwares* que possuem uma única interface de rede. Foi verificado também que o tempo ideal de intervalo entre as medições em hardwares com esta limitação é de 15 segundos. O algoritmo proporciona aos APs uma taxa média de 77% de permanência no melhor canal.

Como trabalhos futuros pretende-se incorporar uma abordagem centralizada a este algoritmo utilizando o modelo de interferência FSI proposto utilizando técnicas de coloração de grafos, agregando a solução maior flexibilidade para tratamentos de seleção de canal em diferentes cenários.

Agradecimentos

Este trabalho foi parcialmente financiado pela Comissão Européia e CNPQ através do projeto de pesquisa IMPRESS (FP7-ICT-2013-EU-Brazil, GA No. 614100), pela FAPEAM através do Projeto de Pesquisa PROTI Amazônia, e pelo Conselho de Pesquisa de Engenharia e Ciências Naturais do Canadá (NSERC) através do Programa NSERC Discovery Grant.

Referências

Alburquerque, Celio Vinicius Neves de. Balbi, H. Souza, F. R. Carrano, Ricardo C. Muchaluat-Saade, D.C. Magalhães, Luiz Claudio Schara (2012) "Algoritmo de seleção de canais centralizado para redes IEEE 802.11 com controlador", II Workshop de Redes de Acesso em Banda Larga - WRA, Brazil, p. 73-86.

Akl, Robert and Arepally, A. (2007). "Dynamic channel assignment in IEEE 802.11 networks", IEEE International Conference on Portable Information Devices, p. 14-18.

Boulmalf, Mohammed and Harroud, Hamid. (2008) "Dynamic Channel Assignment in IEEE 802 .11g", Wireless Communications and Mobile Computing Conference, IWCMC'08 International, pages 864-868.

Burton, Mitchell. (2002) "Channel overlap calculations for 802.11 b networks", White Paper, Cirond Networks Inc.

Chieochan, Surachai. Hossain, Ekram e Diamond, Jeffrey. (2010) "Channel assignment schemes for infrastructure-based 802.11 WLANs: A survey", IEEE Communications Surveys&Tutorials, volume 12, p. 124-136.

Gramacho, S., Araujo, M., and Figueiredo, G. (2013). Dinâmica de seleção de melhores canais em redes IEEE 802.11 com modelo de interferência CCA/SINR. SBRC 2013 / WRA.

Hills, A. (2001). "Large-Scale Wireless LAN Design", IEEE Communications Magazine, p. 98-104.

Mahonen, P., Riihijarvi, J., and Petrova, M. (2004). Automatic channel allocation for small wireless local area networks using graph colouring algorithm approach.15thIEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004, vol.1, p. 536-539.

Mishra and V. Shrivastava, D. Agrawal, S. Banerjee, and S. Ganguly, (2006) "Distributed Channel Management in Uncoordinated Wireless Environments,", in Proc. International Conference on Mobile Computing and Networking, p. 170–181.

Silva, M. W. and Rezende, J. F. (2006). "A Dynamic Channel Allocation Mechanism for IEEE 802.11 Networks", VI International Telecommunications Symposium (ITS 2006), p. 403-408.

Rozner, E. Mehta, Y. Akella, A. Qiu, L. (2012) "Traffic-Aware Channel Assignment in Enterprise Wireless LANs", 2007 IEEE International Conference on Network Protocols", p. 133-143.

Zhou, Haiyan and Liu, Chao (2012) "WLAN Channel Assignment Based on Channel Overlap Factor", Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on, p. 249-251.

Reduzindo os Efeitos do *Bufferbloat* sobre Multi-Caminhos em Redes Sem Fio Heterogêneas

Benevid Felix, Aldri Santos, Michele Nogueira

[1]Núcleo de Redes Sem-Fio e Redes Avançadas (NR2)
Universidade Federal do Paraná (UFPR)

{benevid,aldri,michele}@inf.ufpr.br

Abstract. *Low latency and higher throughput requirements on Internet access have driven advances in order to manage the high-density resulted from the heterogeneous wireless networks (HetNets). The Multi-Path Transmission Control Protocol (MPTCP) allows embedded mobile devices to transmit data over multiple networks by heterogeneous technologies. However, MPTCP yields performance degradation when applied over HetNets, even more under bufferbloat effects. This paper provides a detailed analysis of jointly the MPTCP congestion control mechanism and the AQM CoDel, which has offered promising results against bufferbloat effects. Further, a new discipline, LIFO-CoDel, is proposed aiming to reduce the dropped packets through the MPTCP congestion control mechanism and prioritizing the most recent packets. Simulation results indicate increases in throughput, keeping RTT low, and a reduction in the waiting time.*

Resumo. *A demanda por baixa latência e maior vazão no acesso à Internet tem impulsionado avanços na gerência da alta densidade existente das redes heterogêneas sem fio (HetNets). O protocolo MPTCP (Multi-Path Transmission Control Protocol) possibilita dispositivos móveis embarcados transmitir dados por múltiplas redes através de tecnologias heterogêneas. Entretanto, o MPTCP gera degradações no desempenho em HetNets, ainda mais diante do fenômeno bufferbloat. Este trabalho provê uma análise detalhada do controle de congestionamento do MPTCP em conjunto com a disciplina AQM CoDel, a qual se apresenta como promissora contra os efeitos do fenômeno bufferbloat. Além disso, uma nova disciplina, a LIFO-CoDel, é proposta com o objetivo de reduzir a taxa de pacotes descartados pelo controle de congestionamento do MPTCP em redes heterogêneas sem fio através do mecanismo de controle de congestionamento do MPTCP e da prioridade aos pacotes mais recentes. Resultados de simulações indicam a melhoria da vazão, mantendo o RTT baixo, e a redução do tempo de espera dos pacotes.*

1. Introdução

A demanda pelo acesso à Internet com requisitos de baixa latência e maior largura de banda é crescente, especialmente em redes celulares [Hossain and Hasan 2015]. Uma das soluções para atender a esta demanda consiste na introdução de pequenas células, densificando os pontos de acesso à rede. Isto resulta em uma rede heterogênea sem fio (HetNet - *Heterogeneous Network*) combinando células grandes e pequenas com diferentes tecnologias, desempenho, cobertura e qualidade de serviço [Jeanette Wannstrom 2014].

Os dispositivos móveis embarcados com diferentes tecnologias de acesso sem fio (*Multihomed*) provêem acesso simultâneo a múltiplas redes (Celular, WiFi, etc). Este acesso múltiplo permite o encaminhamento dos dados por diferentes caminhos na tentativa de alcançar objetivos variados, tais como maior vazão, transições mais suaves, e outros.

Devido ao protocolo TCP original não suportar transmissões por múltiplos caminhos, tem-se explorado no contexto de redes celulares o MPTCP, uma versão multi-caminhos do protocolo TCP [Chen et al. 2013, Lee and Lee 2015]. O seu objetivo é manter compatibilidade com o TCP, provendo o envio por multi-caminhos apenas quando os envolvidos na conexão suportam o protocolo. O controle de congestionamento é executado fim-a-fim, como no TCP, e faz uso de sinais como estouro de temporizadores e reconhecimentos (ACKs) duplicados para identificar um possível congestionamento na rede e reduzir o tamanho de sua janela de congestionamento [Raiciu et al. 2012]. O uso do protocolo em servidores ainda é incipiente em ao TCP. Contudo, estudos realizados em [Mehani et al. 2015] demonstraram o interesse dos provedores de serviços em oferecer suporte a conexões multi-caminhos.

Apesar do MPTCP apresentar bom desempenho no contexto de redes de centros de dados [Raiciu et al. 2011, Singh et al. 2015], muitos problemas existem quando aplicado em HetNets [Ferlin et al. 2014, Khalili et al. 2013, Arzani et al. 2014]. O MPTCP sofre degradação no desempenho em cenários com caminhos heterogêneos, com variação na taxa de perda de pacotes, latência e taxa de transferência. Esta degradação aumenta quando os caminhos apresentam grandes variações do atraso causadas pelo fenômeno *bufferbloat* [Ferlin-Oliveira et al. 2014]. Este fenômeno, evidenciado por [Nichols and Jacobson 2012], ocorre devido à existência de grandes filas que absorvem enorme quantidade de tráfego em um canal congestionado, acarretando alta latência com baixa perda de pacotes. Uma vez que o controle de congestionamento do MPTCP utiliza a perda de pacotes como sinal para identificar o congestionamento, o *bufferbloat* não percebido acaba interferindo no desempenho dos demais caminhos.

Estudos recentes acerca do *bufferbloat* atestam a existência do problema em redes sem fio [Alfredsson et al. 2013]. As abordagens empregadas para mitigar o problema envolvem algoritmos de controle de congestionamento (CC) ou disciplinas de controle ativo de filas (AQM - *Active Queue Management*). O primeiro depende do algoritmo de CC fim-a-fim, e o segundo, da implementação de uma disciplina AQM nos pontos de encaminhamento da rede. Estudos realizados em [Chen et al. 2013, Ferlin et al. 2014, Zhou et al. 2015] mostram que os algoritmos de CC do MPTCP sofrem degradações em um cenário de HetNets. No melhor de nosso conhecimento, a disciplina AQM *Controlled Delay* (CoDel) [K. Nichols 2014] e suas variações, como *Flow Queue Controlled Delay* (FQ-CoDel) [Hoeiland-Joergensen et al. 2015], possuem bons resultados frente ao fenômeno com redução expressiva do tempo de permanência na fila.

A fim de compreender o funcionamento destas duas abordagens realizamos simulações utilizando o NS-3 [Kheirkhah 2015] comparando os algoritmos de CC e as disciplinas de fila. Os resultados obtidos indicam que o mecanismo de controle de fila do CoDel causa muitos descartes de pacotes na fila. Isto impacta no CC do MPTCP que acaba não alcançando a capacidade de transferência do canal. A partir dos resultados desta análise, a disciplina LIFO-CoDel foi proposta, a qual altera a lógica da fila, originalmente FIFO para LIFO. O objetivo da alteração é permitir o encaminhamento dos

pacotes com menor tempo de espera. O tempo máximo do pacote na pilha é controlado com um parâmetro *theta* (θ) auto-ajustável. Com isto, o número de descartes é reduzido e consequentemente uma melhora no desempenho da transmissão multi-caminhos. Os resultados das simulações indicam uma melhora na vazão, com redução do número de descartes, e mantendo uma equidade no *Round Trip Time* (RTT) com o FIFO-CoDel no cenário de HetNets abordado neste trabalho.

O restante do trabalho está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha as características do protocolo MPTCP e o funcionamento do CoDel e FQ-CoDel. A Seção 4 apresenta os resultados da análise do impacto do CoDel e FQ-CoDel no CC do MPTCP em um cenário de redes heterogêneas. A Seção 5 descreve a proposta do LIFO-CoDel e os resultados obtidos. A Seção 6 apresenta as conclusões e direções futuras.

2. Trabalhos Relacionados

Diferentes trabalhos como [Ferlin-Oliveira et al. 2014], [Gómez et al. 2014] e [Chen and Towsley 2014] têm avaliado o impacto do *bufferbloat* em conexões MPTCP. Em [Gómez et al. 2014] foram realizadas simulações comparando diferentes algoritmos de CC do MPTCP e diferentes abordagens de roteamento multi-caminho. Em relação ao CC, os resultados mostram que o MPTCP sofre degradação do desempenho sob o aumento na taxa de perda do caminho, sendo o impacto maior quando o protocolo utiliza múltiplas interfaces, cenário comum em redes heterogêneas. O atraso em um dos caminhos impacta no desempenho dos demais e em alguns cenários o desempenho somado dos caminhos pode ser inferior ao de um único fluxo TCP, não justificando o uso do protocolo [Arzani et al. 2014]. Em [Arzani et al. 2014], os autores apresentam resultados obtidos através de experimentos com MPTCP em cenários heterogêneos. Um dos objetivos era compreender o papel da seleção de caminhos no desempenho do MPTCP. Os resultados obtidos demonstraram que o mecanismo de CC do MPTCP não precisa gerenciar a seleção de sub-fluxos e que a escolha do escalonador de pacotes tem grande impacto na vazão final. A solução proposta em [Ferlin et al. 2014] avalia dinamicamente o desempenho dos caminhos para adaptar o escalonamento dos dados. Os resultados apresentados mostraram que a solução é melhor que o escalonador padrão do MPTCP quando os caminhos possuem diferentes características, como redes 2G e WiFi, 2G e 3G. Porém, quando a heterogeneidade do caminho é menor, o desempenho é igual ou inferior ao escalonador padrão.

O trabalho de [Lee and Lee 2015] investiga como diferentes características das redes heterogêneas podem afetar o desempenho do MPTCP. Para melhorar o desempenho do protocolo quando o mesmo apresenta problemas como o bloqueio da fila de recebimento (HOL -*Head of Line*) e *bufferbloat*, um algoritmo de ajuste dinâmico da janela de recebimento do MPTCP com base nos valores dos RTTs é proposto. Contudo, os autores consideram um cenário em que a taxa de perda da rede celular é menor que a rede usando WiFi e dependem de um tempo mínimo de observação do RTT. A variação entre os RTTs dificulta estimar o tamanho da janela de recebimento. Em [Chen and Towsley 2014], os autores analisaram o bloqueio HOL causado pelo impacto do fenômeno *bufferbloat* e demonstraram que o caminho mais rápido passa por diversos períodos de ociosidade. Isto acarreta na degradação do desempenho porque o mecanismo de *idle restart* do MPTCP reinicia a janela em modo de partida lenta (*slow start*). Os autores sugeriram desativar o

mecanismo para que a janela continue crescendo após o período de ociosidade. Apesar de melhorar o desempenho, este ajuste torna os fluxos do MPTCP mais agressivos e podem interferir no objetivo do protocolo de manter o balanceamento com demais fluxos. Em [Ferlin-Oliveira et al. 2014], os autores avaliam os desafios da transmissão de dados através do MPTCP em cenários com caminhos heterogêneos e com a presença do *bufferbloat*. Os resultados mostram que em determinados cenários o MPTCP não consegue ultrapassar o desempenho do melhor caminho.

Apesar de não estar diretamente relacionado à transmissão multi-caminhos, o trabalho realizado em [Kulatunga et al. 2015] avalia o desempenho do CoDel e FQ-CoDel em redes com capacidade de transmissão limitada e com altos valores de RTTs. Os resultados apresentados mostram que o descartes na fila causados pelo CoDel impacta de forma crescente na taxa de transmissão conforme o RTT do caminho aumenta. A proposta do trabalho foi de otimizar os parâmetros do CoDel para equilibrar a taxa de transmissão e reduzir o impacto no CC s efeitos do *bufferbloat* . Apesar dos resultados justificarem a abordagem, é importante destacar que o ajuste de parâmetros pode ser impraticável em cenários de HetNets, nos quais as condições da rede mudam constantemente, e que isto deve ser feito de forma dinâmica.

3. O TCP Multi-caminhos e o Controle de Filas

O protocolo MPTCP [Barré et al. 2011] tem como base o protocolo TCP e possibilita a transmissão de dados por multi-caminhos. Como ilustrado na Figura 1, cada sub-fluxo MPTCP se comporta como um fluxo TCP. Cada sub-fluxo se associa a um endereço de rede e uma interface diferente (Figura 1.a) ou ainda, múltiplos fluxos podem ser criados sobre uma única interface existente (Figura 1.b). Entretanto, uma única conexão fim-a-fim é estabelecida. O protocolo possibilita adicionar ou remover caminhos enquanto a conexão estiver ativa. Alguns dos objetivos do protocolo é obter um desempenho melhor ao oferecido por um único fluxo e aumentar a resiliência utilizando diferentes caminhos [Ford et al. 2013].

a) Múltiplas interfaces *b) Interface única*

Figura 1. Arquitetura MPTCP

O CC executado pelo MPTCP tem como base os algoritmos do TCP, como a variante *NewReno*. Todos eles modificam apenas a fase de prevenção de congestionamento (*Congestion Avoidance*), ficando as fases de inicialização lenta (*Slow Start*), retransmissão rápida (*Fast Retransmit*) e recuperação rápida (*Fast Recovery*) inalteradas [Singh et al. 2013]. O CC é executado em nível de sub-fluxo, utilizando uma janela de congestionamento (CWND) individual e uma janela de recebimento (RCVW) compartilhada para facilitar a entrega ordenada dos dados [Zhou et al. 2015]. Seus requisitos principais incluem aumentar a utilização vazão, não prejudicar fluxos concorrentes

e balancear o congestionamento. A vazão dos sub-fluxos em uma transferência multi-caminhos deve ser pelo menos melhor que um único fluxo através do melhor caminho. Os sub-fluxos precisam se comportar como um único fluxo TCP na existência de um gargalo comum. Por fim, o tráfego deve ser direcionado para os caminhos menos congestionados.

A maioria dos algoritmos de CC propostos para o MPTCP aplicam diferentes abordagens às fases de incremento aditivo e decremento multiplicativo. O algoritmo LIA (*Linked Increase Algorithm*), adotado como padrão pelo IETF [Ford et al. 2013], especifica apenas como deve ocorrer o incremento da janela de congestionamento ao receber um ACK, mantendo o decremento padrão como no TCP. O crescimento das janelas dos sub-fluxos são acoplados (Eq. 1). O parâmetro α, Eq. 2, controla a agressividade dos sub-fluxos de modo que o incremento da janela não seja superior ao de um fluxo de caminho único com o mesmo tamanho de janela. Nas Eqs. 1 e 2, w_i e RTT_i referem-se ao tamanho da janela de congestionamento e o tempo de ida e volta (RTT) de um sub-fluxo i, respectivamente. O parâmetro W representa o tamanho total das janelas de congestionamento de todos os sub-fluxos.

$$w_i = \begin{cases} min(\alpha/W, 1/w_i) & incremento \\ w_i/2 & decremento \end{cases} \tag{1}$$

$$\alpha = W * \frac{max\left(w_i/RTT_i^2\right)}{\left(\sum_i w_i/RTT_i\right)^2} \tag{2}$$

Em HetNets, o MPTCP tem apresentado diversos problemas que degradam seu desempenho. Isto ocorre porque os caminhos possuem diferentes características, como variação da largura de banda, atraso, taxa de perda, entre outras [Ferlin et al. 2014]. Apesar dos problemas, o cenário das HetNets com o MPTCP é muito promissor [Kurosaka and Bandai 2015]. As HetNets têm sido uma das propostas do *3rd Generation Partnership Project* (3GPP) [Jeanette Wannstrom 2014] para atender aos requisitos de serviços cada vez mais restritos, e.g., a 5º geração das redes celulares requerem latência entre $2ms$ a $5ms$ e vazão aproximada de 10 Gbbs [Hossain and Hasan 2015]. Esta proposta inclui a densificação da rede, com a inserção de células de curto alcance (e.g. micro, pico, metro) para ampliar a cobertura, reduzir a latência e aumentar a vazão. Estas pequenas células implementam diferentes tecnologias de acesso, possibilitando que os dispositivos móveis, com suporte a múltiplas interfaces de rede (*multihomed*), possam explorar os benefícios da transmissão multi-caminhos utilizando o protocolo MPTCP.

A maioria dos roteadores atuais empregam um mecanismo FIFO (*First In, First Out*) para encaminhamento dos pacotes, e por simplicidade, descartam aqueles que chegam quando a fila está cheia (*DropTail*). A disciplina *DropTail* frente às grandes filas presentes nos roteadores e a um congestionamento persistente tem acarretado grandes variações no atraso. Sobre a perspectiva deste problema, a disciplina FIFO é ineficiente. O problema das grandes filas resulta no aumento excessivo da latência e degradação da vazão que acabam afetando a disponibilidade da rede. Em [Nichols and Jacobson 2012], este problema é evidenciado como fenômeno *bufferbloat* e uma disciplina denominada *Controlled Delay* (CoDel) é proposta para mitigar o problema. A disciplina CoDel guarda o instante enq_i que cada pacote p_i é colocado na fila. Ao retirar o pacote da fila no tempo

deq_i, ele calcula o tempo de permanência do pacote, conforme Eq. 3).

$$\delta_i = deq_i - enq_i \tag{3}$$

Este intervalo é utilizado pelo mecanismo empregado pelo CoDel para alternar entre dois estados: descarte e não-descarte. A disciplina CoDel inicia no estado de não-descarte e pode mudar seu estado conforme valor do intervalo (μ). O valor inicial de μ é λ, porém o valor é ajustado conforme o número de descartes consecutivos (n_{drop}). Se entre o tempo $t_i = t$ e $t2 = t + \mu$ um pacote p_i é desenfileirado tal que, $deq_i \in [t_1, t_2]$ e $\delta_i > \tau$, a CoDel entra em estado de descarte e os próximos pacotes desenfileirados são descartados enquanto $\delta_i > \tau$. A variável que controla do número de descartes consecutivos é incrementada em $n_{drop} += 1$ e o valor de μ é atualizado para $\mu\sqrt{n_{drop}}$. Caso $\delta_i < \tau$, a disciplina CoDel permanece ou entra no estado de não-descarte e a variável n_{drop} é reinicializada com 1 e μ para λ [Kulatunga et al. 2015].

A variação FQ-CoDel emprega a mesma abordagem do CoDel para controlar o atraso do pacote na fila. A diferença é que ele separa os fluxos em n filas para aplicar o mecanismo do CoDel de forma independente. Para tornar a utilização justa, os pacotes são escalonados com um algoritmo *Deficit Round Robin* (DDR) [Hoeiland-Joergensen et al. 2015]. O algoritmo DDR configura uma quantidade de Q_i (*quantum*) *bytes* para ser retirado da fila n_i a cada ciclo. Caso a fila n_i não tenha Q_i *bytes*, este déficit é acumulado para o próximo ciclo. O CoDel e FQ-CoDel sugerem valores $\tau = 5ms$ e $\lambda = 100ms$ como padrão para maioria das situações [Ali et al. 2015].

4. Análise do Impacto do CoDel no CC do MPTCP em HetNets

A entre o fenômeno *bufferbloat* e o CC do MPTCP, embora realizada por alguns trabalhos citados na seção 2, não consideram o impacto das disciplinas de fila na transmissão multi-caminhos. Uma vez que o MPTCP utiliza um CC acoplado, se faz necessário avaliar a implicação das perdas de pacotes na fila, ocasionadas tanto por congestionamentos, quanto pelo mecanismo empregado pela disciplina de fila. Para avaliar este impacto, foram realizadas simulações com NS-3 [Kheirkhah 2015] considerando um cenário com duas redes de acesso, celular e WiFi, comum para usuários de *smartphones*, ilustrado pela Figura 2. Neste cenário o equipamento do usuário (EU) estabelece a conexão MPTCP com o servidor pela sua interface 1 através da rede LTE (caminho A). Após o passo anterior, um sub-fluxo adicional é estabelecido pela interface 2 através da rede WiFi (caminho B). Ambos os caminhos, A e B, compartilham a fila de saída do roteador que dá acesso ao servidor. O caminho C é configurado intencionalmente com uma vazão inferior aos demais para representar um gargalo na rede. A partir destas definições, o cenário permite representar uma situação em que os sub-fluxos do MPTCP, originados da EU, compartilhem um ponto de congestionamento na (i.e. fila do roteador).

As simulações empregam os fatores listados na Tabela 1. Além desses, os pacotes possuem 1458 bytes, a fim de saturar mais rapidamente a janela de congestionamento, o tamanho máximo da fila é de 50 pacotes, e RCVWD e CWND são definidos com valor padrão do simulador de 64 Kilobytes. A carga de trabalho foi fixada em 4 Mbytes para facilitar o cálculo do *goodput*. E com intuito de não limitar os caminhos A e B até o roteador, a taxa de transferência foi definida com 1 Gbps e atraso padrão de $1ms$, salvo

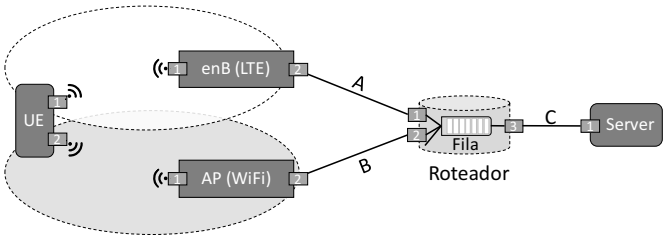


Figura 2. Cenário de Avaliação

os parâmetros definidos na tabela. Estes valores são definidos para que o caminho C funcione como um gargalo na transmissão de dados entre a EU e o servidor. O ponto de acesso (AP) emprega o padrão IEEE802.11a, deixando inalterados os valores padrões das camadas inferiores. Embora a EU tenha permanecido em uma posição fixa durante as simulações, o fator atraso do caminho, definido para o caminho A, tende representar as variações que ocorrem em função da mobilidade. Esta variação apenas foi definida para a rede LTE para representar a heterogeneidade entre os caminhos. E por fim, com objetivo de preencher a fila do roteador, todas as simulações foram realizadas com um terceiro nó transmitindo pacotes UDP para o servidor, ocupando 25% da banda do caminho C. Desta forma, os demais sub-fluxos se adequam aos outros 75% disponíveis.

Tabela 1. Fatores utilizados nas simulações

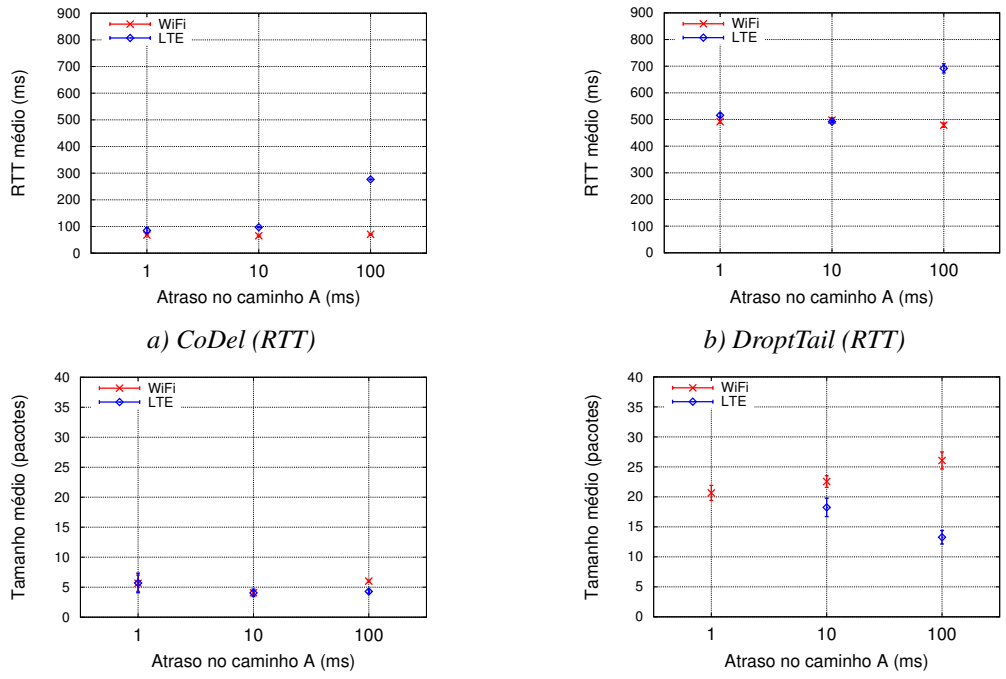
Fator	Objetivo	Valor
Atraso do caminho	Variar o RTT do caminho A	1ms, 10ms, 100ms
Disciplina AQM	Analisar o comportamento da fila	DropTail, CoDel
Algoritmo CC	Analisar o CC do MPTCP	Uncoupled, LIA e RTT_Compensator

Para avaliar o impacto da fila na transmissão de dados foram utilizadas métricas para medir o desempenho da transmissão fim-a-fim, do CC e da fila. Para análise da transmissão foi empregado o *goodput* ($bytes\_recebidos/tempo\_total$). O tamanho da janela de congestionamento e o RTT são considerados para analisar o CC do caminho. E por fim, o número de pacotes descartados, o tamanho e o atraso (tempo de permanência) foram aplicados para medir o desempenho da fila. Estas métricas são apresentadas com valores médios de 35 repetições e com intervalo de confiança de 95%.

4.1. Resultados

Os resultados apresentados referem-se à comparação entre os algoritmos LIA, $RTT\_Compensator$ e $Uncoupled$ com as disciplinas CoDel e DropTail. Esta última foi empregada como base para comparações. A Figura 3 apresenta os resultados do tamanho médio da janela de congestionamento (CWND) e o valor médio do RTT para o CoDel e DropTail, provenientes de simulações realizadas com o algoritmo LIA. Em todos os gráficos, o eixo x representa o atraso atribuído ao caminho A. Com ambos os caminhos com atraso de $1ms$, o RTT médio observado, tendo o CoDel como disciplina de fila do roteador, foi de $85ms$ para o caminho A (LTE) e $67ms$ para o B (WiFi). Os resultados indicam que a variação do atraso do caminho A influencia o RTT do caminho B. Com atraso de $100ms$, o RTT médio do caminho A foi de $276ms$ e o caminho B variou entre

$69ms$ a $71ms$. Tendo como disciplina de fila o DropTail, o congestionamento tem um impacto considerável no RTT médio, atingindo $691ms$ para o caminho A e $478ms$ para o B com atraso de $100ms$. Estes valores são quase 9 vezes maiores que os obtidos pelo CoDel. Por limitação de espaço, os gráficos com os resultados para *RTT_Compensator* e *Uncoupled* não foram apresentados, porém tiveram uma variação em relação ao LIA de aproximadamente 5% a 8%, respectivamente.



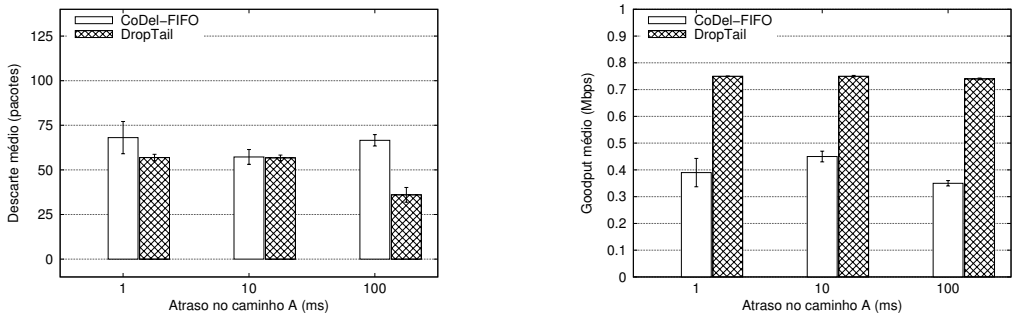
a) CoDel (RTT) *b) DroptTail (RTT)*

c) CoDel (CWND) *d) DropTail (CWND)*

Figura 3. Desempenho da transmissão

Ainda na Figura 3, pode-se observar que o comportamento da janela de congestionamento (CWND) reflete o funcionamento do algoritmo de CC do MPTCP. Conforme o atraso aumenta no caminho A, a sua CWND é reduzida a fim de priorizar a CWND do caminho B, que possui um menor RTT. Contudo, este balanceamento do algoritmo não ocorre de forma adequada quando o CoDel é empregado, o que pode ser observado no gráfico quando o caminho A tem atraso de $10ms$. Neste caso, ambas as janelas tiveram um redução de tamanho e os resultados apresentados na Figura 4.a, referentes ao descarte médio na fila, ajudam a compreender o porquê. O CoDel (CoDel-FIFO) descarta um número muito maior de pacotes que o DropTail, impedindo a CWND de crescer. Os descartes tem grande impacto no desempenho da transmissão multi-caminhos, como pode ser observado no *goodput* apresentado pela Figura 4.b. O DropTail possibilita que toda a capacidade do canal disponível para as transmissões com MPTCP seja alcançada (i.e. 75%), enquanto o CoDel um pouco mais de 50%.

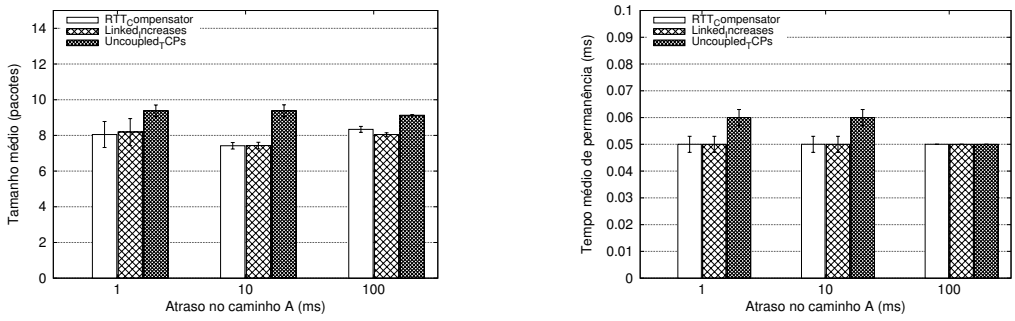
A variação do *goodput* do CoDel está diretamente relacionado ao número de descartes de pacotes na fila. Conforme o número de pacotes descartados aumenta, diminui o *goodput* da transmissão. Contudo, o RTT médio é expressivamente menor que o observado com o DropTail. Como o DropTail não descarta pacotes enquanto a fila é suficiente para absorver o tráfego, este atraso acarreta um RTT médio muito alto, no entanto, como



a) *b)*

Figura 4. Descartes vs. Goodput

a fila está sempre cheia, a utilização do canal é mais eficiente. Com o CoDel, a fila não atinge 20% de sua capacidade. Isto pode ser observado nos resultados apresentados na Figura 5.a, referentes ao tamanho médio da fila alcançado com os diferentes algoritmos de CC. O algoritmo *Uncoupled* possui resultados mais divergentes com o algoritmo padrão (LIA) porque não acopla os sub-fluxos. Deste modo o atraso no caminho A interfere menos no caminho B e o mesmo consegue encaminhar mais pacotes. Mas isto também interfere no tempo de permanência do pacote na fila. Conforme Figura 5.b, este tempo é maior que os demais na maioria dos casos observados.



a) *b)*

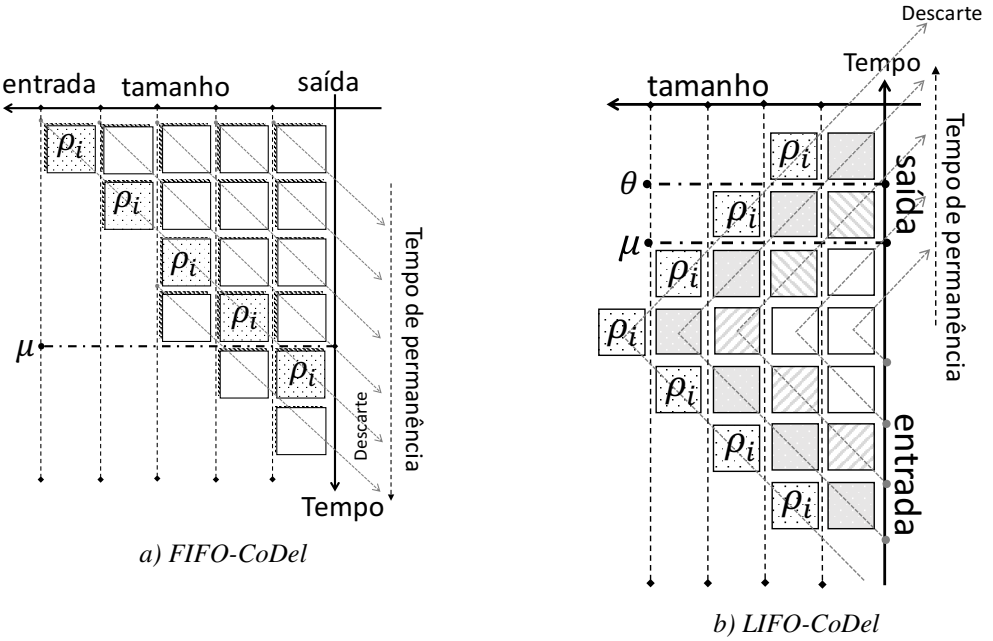
Figura 5. Estado da fila com CoDel vs. MPTCP CC

O CoDel, a partir deste ponto denominado FIFO-CoDel, reduz drasticamente o RTT dos caminhos em relação à disciplina padrão DropTail, contudo, penaliza o *goodput* da transmissão devido ao elevado número de descartes na fila. Este *trade-off* entre o *goodput* e a latência é percebido desde a proposta inicial da disicplina FIFO-CoDel em [Nichols and Jacobson 2012], porém com o MPTCP, se demonstra mais expressivo. Com intuito de reduzir este *tradeoff*, a próxima seção apresenta o LIFO-CoDel.

5. LIFO-CoDel: Um Nova Disciplina de Fila

Esta seção apresenta LIFO-CoDel, uma nova proposta para reduzir o impacto causado pelo mecanismo de controle de fila no CC do MPTCP. A estratégia empregada pela proposta altera o escalonamento dos pacotes FIFO (*First In, First Out*), utilizada originalmente pelo FIFO-CoDel, para LIFO (*Last In, First Out*) além dos demais ajustes detalhados a seguir. Esta proposta tem seus fundamentos no trabalho de [Maurer 2015] que

utiliza uma estratégia denominada LIFO adaptativo com CoDel para controlar a sobrecarga de requisições nos servidores do Facebook. Neste trabalho, a proposta LIFO-CoDel é projetada para um contexto original de redes heterogêneas sem fio, as quais precisam de soluções específicas capazes de tratar suas características peculiares em conjunto com os requisitos de suas aplicações.



a) FIFO-CoDel

b) LIFO-CoDel

Figura 6. Atraso na fila vs. pilha

A fim de mostrar a diferença entre FIFO e LIFO, a Figura 6 ilustra em um plano cartesiano de tamanho da fila vs. tempo, a entrada, permanência e saída de pacotes das estruturas fila e pilha. Na disciplina FIFO-CoDel, conforme a fila cresce, o atraso do pacote aumenta. O parâmetro τ controla o atraso do pacote e o μ o tamanho da fila. Quando o pacote ρ_i é desenfileirado ele somente é descartado se δ_i é maior que τ e o instante $t = deq_i$ for maior que μ. Este último se ajusta dinamicamente conforme o número de descartes para controlar o tamanho da fila. Isto permite que pacotes com um tempo de permanência δ_i maiores que τ sejam encaminhados até que se entre novamente em estado de descarte. Contudo, na seção 4.1, os resultados indicam que este parâmetro não é efetivo. Este mecanismo não controla o tamanho da fila de modo que o canal seja ocupado de forma eficiente. Por este motivo LIFO-CoDel propõe duas principais alterações: alterar a fila para pilha, a fim de priorizar os pacotes com menor tempo de permanência; um novo parâmetro para controlar os descartes na fila.

A alteração da estrutura fila para pilha permite priorizar a retirada dos pacotes com menor tempo de permanência, ou seja os pacotes mais recentes. Considerando que o mecanismo esteja em estando de descarte, na Figura 6.b o pacote p_i seria descartado porque seu tempo de permanência é maior que τ e $deq_i > \mu$. Contudo, LIFO-CoDel utiliza o parâmetro θ para determinar se o pacote será descartado. O parâmetro θ representa a razão entre o tempo de permanência máximo δ_{max} (Eq. 4) e o tempo de permanência médio $\bar{\delta}$ (Eq. 5) para o n-ésimo pacote retirado da pilha, dado pela Eq. 6. Se $\bar{\delta} = 0$, então se assume $\theta = 0$. Juntamente com o θ, é calculado Γ_t, que se refere a diferença entre o

tempo de permanência atual e o anterior (Eq. 7). O valor de Γ_t é utilizado para incrementar ou reiniciar o valor da variável k (Eq. 8). Quando $\Gamma_t > 0$, assumindo a estrutura pilha, refere-se a um pacote já na pilha antes da última retirada, porém pode indicar também um atraso maior do sistema em relação à última retirada. No entanto, o valor de k e θ é reinicializado quando $\Gamma_t < 0$, ou seja, pacote atual possui um tempo de permanência menor que o anterior. Por fim, o pacote p_i somente é descartado se $k > \theta$.

$$\delta_{max} = max(\delta_1, \ldots, \delta_n) \tag{4}$$

$$\bar{\delta} = \frac{1}{n}\sum_{i=1}^{n}\delta_i \tag{5}$$

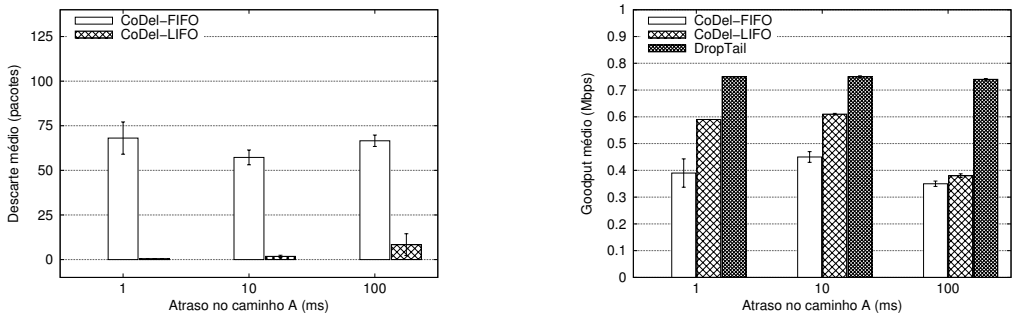
$$\theta = \frac{\delta_{max}}{\bar{\delta}}, \bar{\delta} \neq 0 \tag{6}$$

$$\Gamma_t = \delta_i - \delta_{i-1} \tag{7}$$

$$k = \begin{cases} k+1, & if \quad \Gamma_t > 0 \\ 0, & if \quad \Gamma_t \leq 0 \end{cases} \tag{8}$$

5.1. Resultados

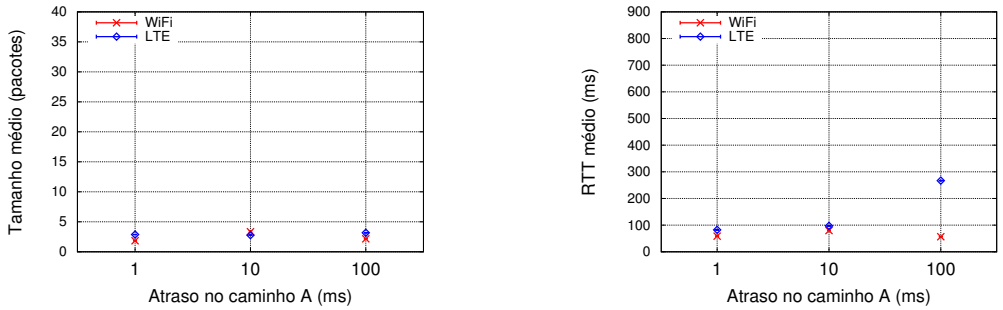
Nesta seção apresentamos os resultados da avaliação da LIFO-CoDel em comparação com os resultados do CoDel e FQ-CoDel. O cenário, as métricas e os parâmetros utilizados nas simulações são os mesmos definidos na Seção 4. Os resultados mostram que o LIFO-CoDel causa um impacto menor no CC do MPTCP, e com isto, melhora o desempenho da transmissão multi-caminhos. Um dos motivos desta melhoria é a redução no número de descartes na fila, ilustrado pela Figura 7.a. Esta redução contribui com o aumento do goodput, conforme Figura 7.b. O LIFO-CoDel tem um *goodput* cerca de 40% maior que FIFO-CoDel, para atraso de $1ms$ no caminho A. O aumento do atraso ocasiona um descarte maior de pacotes, e isto reduz o *goodput* do LIFO-CoDel. Contudo, com atraso de $100ms$ os resultados são discretamente melhores que o FIFO-CoDel.



a) *b)*

Figura 7. Descartes vs. Goodput

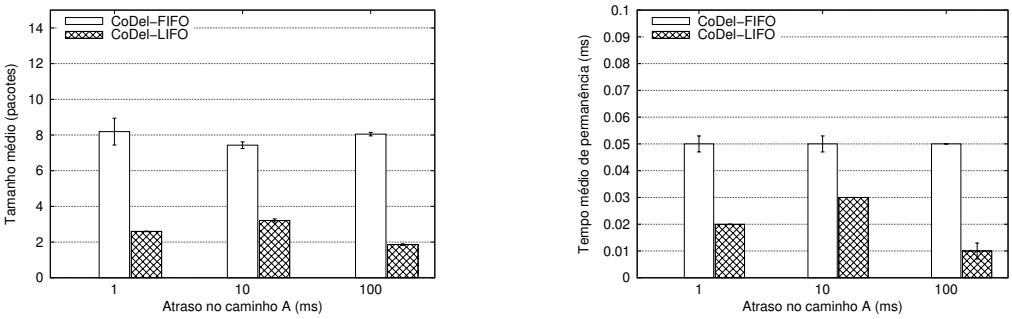
A Figura 8 apresenta os resultados para a variação do tamanho da CWND e RTT do LIFO-CoDel. Assim como CoDel, a janela de congestionamento se manteve em torno de 5 pacotes. Porém, com atraso de $100ms$. O algoritmo CC não priorizou o caminho com menor RTT, como observado nos resultados diante do CoDel e DropTail.



a) CWND *b) RTT*

Figura 8. LIFO-CoDEL com LIA

As Figuras 9.a e 9.b apresentam respectivamente o tamanho médio e o tempo médio de permanência na fila sob as disciplinas FIFO-CoDel e LIFO-CoDel. O LIFO-CoDel reduz o tamanho da fila e consequentemente o tempo de permanência. Este resultados se referem às simulações com o algoritmo LIA. Com o algoritmo $RT\_Compensator$ os resultados variam de 2% a 5%. Com o algoritmo $Uncoupled$, a variação é mais expressiva para o LIFO-CoDel, com valores até 50% acima dos apresentados com LIA.



a) *b)*

Figura 9. Estado da fila

A fim de compreender o impacto do FIFO e LIFO em cada caminho, realizamos simulações com o FQ-CoDel, que separa os sub-fluxos em filas independentes. Os resultados obtidos para ambas as disciplinas tiveram uma variação muito discreta, para o *goodput* ficou entre 2% a 3%. Este resultados indicam que a separação em filas distintas não muda o fato que ambos os sub-fluxos compartilham o mesmo gargalo.

6. Conclusões

Este trabalho analisou o impacto de disciplinas AQM no CC do MPTCP. Os resultados comparando as disciplinas DropTail, CoDel e FQ-CoDel em relação aos algoritmos *Uncoupled*, LIA e *RTT_Compensator* demonstram que existe um *tradeoff* entre RTT e vazão. Diante de *buffers* de grande capacidade, o DropTail não descarta pacotes e a fila cresce acarretando um alto valor de RTT (fenômeno *bufferbloat*). Porém a transmissão alcança a utilização máxima do canal. O CoDel resolve o problema dos atrasos, porém, os descartes impactam no CC do MPTCP e isto reduz a utilização do canal de transmissão. No cenário de redes heterogêneas sem fio, este aspecto tem impacto ainda maior. Ao aumentar o atraso em um dos caminhos se observou uma redução na vazão e aumento

do RTT em ambos os caminhos. Com base nestas observações, este trabalho apresentou a proposta de uma nova disciplina de filas FIFO-CoDel, qual reduz o impacto causado pelo mecanismo de controle de fila no CC do MPTCP com base na disciplina CoDel em cenários de redes heterogêneas sem fio. Os resultados da sua avaliação mostram que a redução no número de descartes na fila contribuo para uma melhoria em métricas como *goodput*, RTT e tempo de permanência na fila.

Como trabalhos futuros, pretende-se avaliar novos algoritmos de CC propostos para o MPTCP no cenário de redes heterogêneas sem fio. Adicionalmente, incluir avaliações que considerem as perdas em decorrência da mobilidade e transição vertical entre as redes. Por fim, ajustar e formalizar matematicamente o parâmetro θ, utilizado pelo LIFO-CoDel, para obter uma melhor resposta em cenários com caminhos heterogêneos que apresentem um atraso acima de $100ms$..

Referências

Alfredsson, S., Del Giudice, G., Garcia, J., Brunstrom, A., De Cicco, L., and Mascolo, S. (2013). Impact of TCP congestion control on bufferbloat in cellular networks. In *IEEE WoWMoM*, páginas 1–7.

Ali, H., Khan, S., and Quaid, M. (2015). Comparative analysis of controlled delay (CoDel) with Deficit Round Robin (DRR) to overcome bufferbloat problem in wired network. *International Journal of Current Engineering and Technology*, 5(5):3378–3386.

Arzani, B., Gurney, A., Cheng, S., Guerin, R., and Loo, B. T. (2014). Impact of path characteristics and scheduling policies on MPTCP performance. In *IEEE WAINA*, páginas 743–748.

Barré, S., Bonaventure, O., Raiciu, C., and Handley, M. (2011). Experimenting with multipath TCP. *ACM SIGCOMM*, 41(4):443–444.

Chen, Y.-C., Lim, Y.-s., Gibbens, R. J., Nahum, E. M., Khalili, R., and Towsley, D. (2013). A measurement-based study of multipath TCP performance over wireless networks. In *ACM IMC*, páginas 455–468.

Chen, Y.-C. and Towsley, D. (2014). On bufferbloat and delay analysis of multipath TCP in wireless networks. In *IEEE IFIP*, páginas 1–9.

Ferlin, S., Dreibholz, T., and Alay, O. (2014). Multi-path transport over heterogeneous wireless networks: Does it really pay off? In *IEEE GLOBECOM*, páginas 4807–4813.

Ferlin-Oliveira, S., Dreibholz, T., and Alay, O. (2014). Tackling the challenge of bufferbloat in multi-path transport over heterogeneous wireless networks. In *IEEE IWQoS*, páginas 123–128.

Ford, A., Raiciu, C., Handley, M., and Bonaventure, O. (2013). TCP extensions for multipath operation with multiple addresses. RFC 6824, IETF.

Gómez, D., Garrido, P., Rabadan, C., Agüero, R., and Muñoz, L. (2014). TCP performance enhancement over wireless mesh networks by means of the combination of multi-RAT devices and the MPTCP protocol. *Network Protocols and Algorithms*, 6(3):56–81.

Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., and Dumazet, E. (2015). FlowQueue-CoDel. Internet-Draft draft-hoeiland-joergensen-aqm-fq-codel-00.txt, IETF.

Hossain, E. and Hasan, M. (2015). 5G cellular: key enabling technologies and research challenges. *IEEE Instrumentation & Measurement Magazine*, 18(3):11–21.

Jeanette Wannstrom, K. M. (2014). HetNet, Small Cells. TS 1576, 3rd Generation Partnership Project (3GPP).

K. Nichols, V. J. (2014). Controlled Delay Active Queue management. RFC draft 6824, IETF.

Khalili, R., Gast, N., Popovic, M., and Le Boudec, J.-Y. (2013). MPTCP is not pareto-optimal: Performance issues and a possible solution. *IEEE/ACM Networking*, 21(5):1651–1665.

Kheirkhah, M. (2015). Multipath TCP in NS-3. Código fonte do projeto disponível em http://dx.doi.org/10.5281/zenodo.32691.

Kulatunga, C., Kuhn, N., Fairhurst, G., and Ros, D. (2015). Tackling bufferbloat in capacity-limited networks. In *IEEE EuCNC*, páginas 381–385.

Kurosaka, T. and Bandai, M. (2015). Multipath TCP with multiple ACKs for heterogeneous communication links. In *IEEE CCNC*, páginas 613–614.

Lee, J. S. and Lee, J. (2015). Multipath TCP performance improvement in mobile network. In *IEEE ICUFN*, páginas 710–714.

Maurer, B. (2015). Fail at scale. *ACM Queue*, 13(8):30:30–30:46.

Mehani, O., Holz, R., Ferlin, S., and Boreli, R. (2015). An early look at multipath TCP deployment in the wild. In *ACM HotPlanet*, páginas 7–12.

Nichols, K. and Jacobson, V. (2012). Controlling Queue Delay. *ACM Queue*, 10(5):20:20–20:34.

Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. (2011). Improving datacenter performance and robustness with multipath TCP. *ACM SIGCOMM*, 41(4):266–277.

Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and Handley, M. (2012). How hard can it be? designing and implementing a deployable multipath TCP. In *USENIX*, páginas 29–29.

Singh, A., Xiang, M., Konsgen, A., Goerg, C., and Zaki, Y. (2013). Enhancing fairness and congestion control in multipath TCP. In *IEEE WMNC*, páginas 1–8.

Singh, S. K., Das, T., and Jukan, A. (2015). A survey on internet multipath routing and provisioning. *IEEE Communications Surveys & Tutorials*, 17(4):2157–2175.

Zhou, D., Song, W., Wang, P., and Zhuang, W. (2015). Multipath TCP for user cooperation in LTE networks. *IEEE Network*, 29(1):18–24.

LMT-MAC: um protocolo MAC multicanal livre de colisões para Redes de Sensores Sem Fio

Gilson M. Júnior[1], **Ariel F. F. Marques**[1] **Luiz H. A. Correia**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA)
Caixa Postal 3037 – 37200-000 – Lavras – MG – Brazil

{gilsonmj,arielmarques}@posgrad.ufla.br, lcorreia@dcc.ufla.br

Abstract. *The protocol for the Medium Access Control (MAC) layer directly influences the operation of Wireless Sensor Networks (WSN), impacting on multiple metrics as latency, throughput and delivery rate. In current literature, multichannel scheduled protocols use complex mechanisms for time synchronization, channel and slots allocation, unsuitable for devices with limited hardware. In this paper, we present the Lightweight Multichannel Tree MAC (LMT-MAC), an efficient MAC protocol based on Time Division Multiple Access (TDMA), multichannel and contention-free. Results obtained by extensive simulations show that LMT-MAC is more effective than other traditional protocols and its implementation in real sensor nodes demonstrate its feasibility.*

Resumo. *O protocolo da camada de controle de acesso ao meio (MAC) influencia diretamente na operação das Redes de Sensores Sem Fio (RSSF), impactando em diversas métricas como latência, vazão e taxa de entrega. Na literatura atual, protocolos TDMA multicanal usam mecanismos complexos para sincronização de tempo, alocação de canais e slots, geralmente inviáveis para dispositivos com hardware limitado. Neste trabalho, é proposto o Lightweight Multichannel Tree MAC (LMT-MAC), um protocolo MAC eficiente, baseado em TDMA, multicanal e livre de contenção. Resultados obtidos por meio de simulações mostram que o LMT-MAC é mais eficiente que outros protocolos tradicionais, e sua implementação em nós reais demonstram sua viabilidade.*

1. Introdução

As Redes de Sensores Sem Fio (RSSF) têm sido empregadas em diversos segmentos para auxiliar no monitoramento e controle de processos, rastreamento de objetos, avaliação de condições do solo para agricultura, monitoramento de infraestrutura, dentre outras aplicações [Akyildiz et al. 2002]. Cada aplicação possui requisitos específicos, os quais determinam as características dos nós sensores e os protocolos de comunicação que devem ser utilizados.

As primeiras aplicações de RSSF tratavam principalmente de monitoramento ambiental, e apenas poucos projetos levavam em consideração requisitos de qualidade de serviço (QoS) para aplicações de tempo real [Romer and Mattern 2004]. Com o desenvolvimento de dispositivos e protocolos mais robustos para comunicação sem fio, soluções baseadas em RSSF para aplicações que demandam maior confiabilidade começaram a emergir, visando oferecer formas de monitoramento mais flexíveis e com custo inferior ao de soluções cabeadas, tradicionalmente utilizadas [Gungor and Hancke 2009].

Redes para missões críticas (*Mission Critical Networks*) são empregadas para aplicações em que uma ou mais características devem ser priorizadas, a fim de satisfazer requisitos de qualidade de serviço. Essas redes diferem tanto de redes comuns, usadas para a conexão de computadores, quanto entre si. Diferentes aplicações demandam a priorização de diferentes requisitos, os quais são fortemente determinados pelo protocolo MAC. Alguns dos requisitos mais comuns a serem priorizados por uma RSSF incluem: energia, vazão de dados (capacidade de entrega de pacotes para os nós), latência, taxa de entrega, resiliência (capacidade da rede se manter operante quando submetida a interferências) e adaptabilidade (reorganização da rede quando ocorrem mudanças de topologia devido a movimentação ou falhas de nós).

Os protocolos MAC propostos inicialmente para RSSF geralmente operavam em apenas um canal, em muitos casos por decisão de projeto e em outros por limitações dos transceptores usados. Como em várias aplicações o foco principal era em economia de energia, fatores como latência e vazão ficavam em segundo plano. Com o suporte a mais canais de comunicação, protocolos multicanais têm sido propostos como forma de aumentar a quantidade de transmissões simultâneas, e, assim, reduzir a latência e aumentar a vazão das redes. Por outro lado, protocolos multicanais apresentam novos desafios como a coordenação de comunicações, atribuição de canais, prevenção de colisões, dentre outros [Incel 2011]. Em protocolos TDMA, a alocação de canais e a distribuição de *slots* tem custo computacional elevado, sendo comum o uso de heurísticas para fazer essa alocação de forma centralizada, geralmente no nó *sink*, que depois é propagada aos demais nós da rede. Essa solução, no entanto, pode ser inviável para RSSF em que os nós possuem recursos limitados.

Em RSSF, o protocolo MAC é apenas parte dos vários softwares em execução nos nós sensores, que podem executar outras tarefas como leitura de sensores, agregação e compressão de dados, comunicação cabeada com outros equipamentos e acionamento de atuadores. Por isso, os recursos consumidos por esse protocolo devem ser minimizados, a fim de causar o menor impacto possível na execução das demais operações.

Neste trabalho é proposto o *Lightweight Multichannel Tree MAC* (LMT-MAC), um protocolo MAC voltado para RSSF compostas por nós cujos recursos de hardware são limitados. O protocolo é focado para aplicações que requerem baixa latência, principalmente naquelas em que é necessário estimar o tempo máximo em que os dados coletados serão enviados até o nó de destino. A configuração dos nós sensores pode ser feita de modo *offline* (para redes planejadas), salvando as configurações dos nós na memória interna de cada um (dispensando o uso de um protocolo específico para a configuração) ou com o uso de um protocolo para configuração em redes não planejadas ou com mobilidade. Com o LMT-MAC, diferentes nós da RSSF podem se comunicar simultaneamente utilizando canais de comunicação diferentes, permitindo obter uma alta vazão de dados. Além disso, o protocolo emprega *buffers* para que os pacotes sejam armazenados até que os nós tenham oportunidade de transmissão, com o objetivo de oferecer também uma alta taxa de entrega.

O trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta em detalhes o protocolo LMT-MAC. A Seção 4 traz a comparação do protocolo com o TreeMAC, e os resultados obtidos com a implementação em nós reais. Por fim, a Seção 5 apresenta as conclusões.

2. Trabalhos Relacionados

Com o foco do LMT-MAC em alta vazão, baixa latência e alta taxa de entrega, nesta seção são analisados protocolos que priorizam algum desses requisitos.

Visando obter baixa latência, o *Real Time Hybrid MAC* (RTH-MAC) [Abdeli et al. 2013] emprega um esquema TDMA centralizado, configurado por uma estação base. Após uma fase inicial de descoberta de vizinhos, a estação base executa um algoritmo para alocação de *slots* e canais para cada nó. O algoritmo de agendamento, baseado em coloração de vértices, faz com que vizinhos a dois saltos de distância utilizem uma tupla (*slot*, canal) diferente. A avaliação do protocolo foi realizada por simulações em MATLAB, e os resultados mostram que o protocolo foi capaz de enviar pacotes através de 5 saltos com uma latência fim-a-fim inferior a 20 milissegundos. O tamanho dos *slots* é calculado pela divisão do tamanho do pacote pela taxa de transferência máxima nominal do transceptor, e os nós são considerados perfeitamente sincronizados após a primeira transmissão, sendo dispensado o uso de períodos de guarda para superar os erros de sincronização. Com essas assunções, é possível definir um *slot* de aproximadamente 1,5 milissegundos (no pior caso, quando são usadas mensagens de confirmação), e, portanto, o protocolo atinge uma latência nas simulações que dificilmente seria possível na implementação em nós reais.

Voltado para aplicações industriais, o PriorityMAC [Shen et al. 2014] é um protocolo híbrido de CSMA e TDMA que emprega classificação de mensagens, dando prioridade mais alta para mensagens críticas e usando métodos de acesso diferentes para tráfego de maior prioridade. O protocolo permite obter latências inferiores ao protocolo MAC do padrão WirelessHART [HART 2015], mas ainda possui problemas quando há contenção entre pacotes de alta prioridade. Se pacotes em nós diferentes são classificados com a mesma prioridade, eles serão transmitidos usando o mesmo modo de acesso ao meio, o que causará colisões e irá gerar atraso justamente na transmissão das informações mais críticas. O PriorityMAC não tira proveito dos recursos de multicanal por usar apenas um canal de comunicação para toda a rede.

O MC-LMAC [Incel et al. 2011] estende o protocolo LMAC para usar múltiplos canais de comunicação. Na configuração inicial da rede são determinados os *slots* e canais que cada nó utilizará para transmitir seus dados. A quantidade de *slots* depende da densidade da rede, sendo determinada durante a inicialização. Os *slots* são divididos em duas partes: a primeira contém vários *subslots*, um para cada canal utilizado. Em cada *subslot*, os nós ouvem no canal relativo ao *subslot* e aguardam que o detentor do *slot* envie uma mensagem de controle, a qual contém o endereço de destino do pacote de dados. Na segunda parte do *slot*, o transmissor e o nó de destino se configuram para o mesmo canal, e os pacotes de dados são transmitidos. A previsibilidade do protocolo é prejudicada no caso da ocorrência de disputas, quando dois transmissores tentam enviar dados ao mesmo nó. Ambos transmissores enviam mensagens informando suas transmissões em seus respectivos *subslots*, no entanto, na fase de transmissão de dados, o nó receptor deve escolher de qual transmissor irá receber por meio de algum mecanismo de prioridade. O outro transmissor deverá aguardar seu próximo *slot* para tentar enviar seus dados. Assim, a transmissão desses dados estará sujeita ao mecanismo usado para priorização, o que dificulta a predição de quando os dados poderão ser transmitidos. Além disso, com a divisão da parte inicial dos *slots* em um *subslot* para cada canal, é necessário aumentar o

tamanho do *slot* para acomodar a seção de transmissão de dados, prejudicando a latência, ou essa seção de dados será menor, prejudicando a vazão.

O TreeMAC [Song et al. 2009] é um protocolo TDMA que organiza a rede como uma árvore e divide um ciclo em *frames*, e cada *frame* em três *slots*. Os nós calculam seu *slot* de comunicação dentro de um *frame* de acordo com a sua profundidade na árvore, e a distribuição de *frames* para cada nó é baseada na quantidade total de filhos que o nó possui. Assim, nós com mais filhos possuem mais oportunidades de transmissão. No TreeMAC, contudo, a transmissão de dados é unidirecional, sendo um protocolo apenas para a coleta de dados, e usa somente um canal para toda a rede. A configuração da rede é feita por meio de um protocolo baseado em contenção para a descoberta de vizinhança. Ao obter a topologia, os *frames* são distribuídos de acordo com a necessidade de cada nó. Após a configuração da rede, cada nó tem conhecimento dos períodos em que deve se manter ativo e dos períodos em que pode desativar o transceptor. Isso reduz a necessidade de mensagens de controle, além de aumentar a previsibilidade do protocolo por evitar disputas por *slots* de transmissão.

A análise dos protocolos MAC propostos atualmente para redes críticas indica a viabilidade de um novo protocolo, que integre diferentes mecanismos para oferecer qualidade de serviço, e que use de forma inteligente os recursos disponíveis nos dispositivos. Além disso, deve considerar dispositivos com restrições de recursos, usando métodos simplificados de configuração.

3. LMT-MAC

O LMT-MAC foi desenvolvido com base no TreeMAC [Song et al. 2009]. Porém, emprega mensagens de controle para permitir comunicação bidirecional, tanto coleta de dados, quanto envio de comandos de atuação de um servidor. Além disso, o LMT-MAC permite comunicação em múltiplos canais, utilizando um esquema simplificado de alocação de canais e *slots*, baseado na distância para o *sink* e na quantidade de nós na sub-rede.

A rede é organizada logicamente como uma árvore, tendo o *sink* como raiz. A configuração dos nós pode ser feita de forma manual, especificando diretamente o endereçamento, a relação entre os nós e os *slots* destinados a cada nó, ou por meio de um protocolo de descoberta de vizinhos baseado em contenção quando a rede é iniciada, como ocorre no TreeMAC. O protocolo de descoberta deve ser capaz de identificar a vizinhança de todos os nós e, então, determinar a árvore geradora mínima da rede. Um período de tempo, ou ciclo, é dividido em vários *frames*, e cada *frame* em dois *slots*, como mostrado na Figura 1.

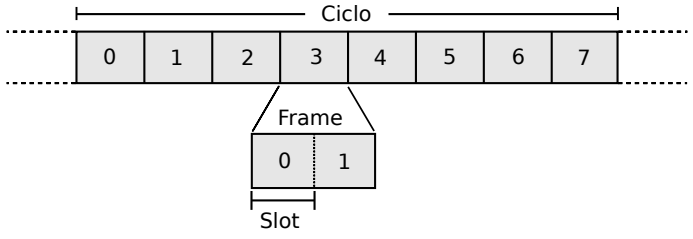


Figura 1. Divisão de tempo usada pelo LMT-MAC.

Para auxiliar na descrição do LMT-MAC, a seguinte notação será adotada:

- N: nós da rede.
- f: canais de comunicação usados.
- F_u: conjunto de *frames* atribuídos ao nó u.
- C_u: conjunto de nós filhos do nó u.
- C_u^k: k-ésimo filho do nó u.
- d_u: profundidade do nó u na árvore (saltos para o *sink*).
- T_{slot}: tempo de duração de um *slot*.
- $F_{LMT-MAC}$: conjunto de todos os *frames* da rede.

Os nós recebem uma quantidade de *frames* proporcional à quantidade de nós em sua sub-rede, ou seja, se um nó tem dois nós filhos em sua sub-rede, ele deve receber dois *frames* (um para cada filho), e um *frame* adicional para a transmissão dos pacotes que não puderam ser transmitidos durante os *frames* anteriores, ou que foram gerados pelo próprio nó. Assim, a quantidade total de *frames* na rede é $|F_{LMT-MAC}| = |N| - 1$, pois o *sink* não precisa de um *frame* adicional. Os *frames* recebidos por um nó i são um subconjunto dos *frames* de seu nó pai u, de modo que $F_{C_u^i} \subseteq F_u$. Para evitar colisões entre nós filhos, cada um recebe um conjunto disjunto de *frames*, de modo que $F_{C_u^i} \cap F_{C_u^j} = \emptyset \ (se\ i \neq j)$. Além disso, os *frames* de um nó são sempre contíguos, de modo que cada nó necessite saber apenas o número do seu *frame* incial, do seu *frame* final e a quantidade total de *frames* na rede. Os nós podem desativar o transceptor para economizar energia quando não estão no seu período ativo (fora de seus *frames*).

Com o conhecimento de seus *frames*, os nós devem definir seus *slots* de transmissão e canais de operação. Um nó u calcula seu *slot* de transmissão usando a Equação 1, e pode calcular o *slot* de seus nós filhos pela Equação 2. O *slot* de transmissão de um nó u é dado por S_u e pode assumir os valores 0 ou 1, indicando um *slot* dos *frames*.

$$S_u \in \{0, 1\} = (d_u - 1) \bmod 2 \tag{1}$$

$$S_{C_u} \in \{0, 1\} = d_u \bmod 2 \tag{2}$$

Após calcular em qual *slot* devem transmitir, os nós determinam em qual canal devem operar em cada *slot*. O número de configurações varia de acordo com a quantidade de frequências utilizadas, tal que $configurações = |f| \times 2$, onde $|f|$ é a quantidade de canais e 2 devido ao uso de dois *slots* por *frame*. Assim, com três canais existem seis configurações diferentes, que os nós usam de acordo com sua profundidade. A Tabela 1 apresenta as configurações para uma rede configurada para usar três canais, de acordo com a profundidade de um nó u. Os valores dos canais podem ser usados como índice para frequências do transceptor, permitindo o uso de canais com intervalos maiores.

Tabela 1. Configurações para uma rede com três canais.

$d_u \bmod 6$	Canal no *slot* 0	Canal no *slot* 1
0	0	2
1	0	0
2	1	0
3	1	1
4	2	1
5	2	2

A transmissão de dados é sempre iniciada pelo nó filho, que verifica em seu *buffer* a existência de pacotes que devem ser encaminhados para o nó pai, e os transmite. Ao

terminar o envio, o nó filho transmite um pacote de controle informando que está pronto para receber pacotes, e o nó pai, por sua vez, transmite os pacotes em buffer destinados ao filho. Ao finalizar as transmissões, o nó pai envia uma mensagem de controle ao nó filho, e ambos podem desativar seus transceptores pelo restante do *slot*.

Nas mensagens de controle trocadas entre os nós são transmitidas as informações necessárias para configuração dos nós e sincronização de tempo. Para configuração dos nós são transmitidas: quantidade de *frames* total, quantidade de *frames* do nó, *frame* inferior e número de saltos para o *sink*. Para sincronização são enviados: tempo global da rede, *frame* atual, *slot* atual e início do próximo *slot*. O LMT-MAC utiliza sincronização global de tempo, na qual o valor do relógio interno do *sink* é tido como referência, e os demais nós da rede devem estimar esse valor. O protocolo de sincronização deve ser capaz de estimar o tempo global a partir das informações do pacote de controle, e sinalizar quando o período de um *slot* se esgotou. Assim o LMT-MAC atualiza seus contadores de *frame* e *slot*, e se configura para o canal apropriado para realizar a transmissão ou recepção de dados. A troca de canais em transceptores comumente usados em nós sensores, como CC2420 e nRF24l01+, é inferior a 1 milissegundo [Instruments 2006][Nordic_Semiconductor 2013], e, portanto, seu impacto pode ser desconsiderado ou acomodado dentro do período de guarda dos *slots*.

O LMT-MAC é desacoplado do protocolo de sincronização, e com protocolos mais sofisticados, pode-se obter uma sincronização mais precisa, permitindo reduzir os tempos de *slot* e guarda, ao custo de algoritmos que requerem mais recursos de hardware. Assim, a escolha do protocolo de sincronização depende dos nós sensores utilizados na RSSF e de decisões de projeto. Ainda assim, no início dos *slots* são definidos períodos de guarda para evitar erros de transmissão devido a imprecisão da sincronização. A avaliação do protocolo de sincronização está fora do escopo deste trabalho.

Cada nó é capaz de estimar a latência máxima de seus pacotes para o *sink*, com base nos valores de T_{slot} (duração de *slot*), d_u (profundidade do nó), $|F_{LMT-MAC}|$ (quantidade total de *frames*) e $|F_u|$ (quantidade de *frames* pertencentes ao nó), usando a Equação 3. A equação considera que o período de um *slot* é suficiente para a transmissão de todos os pacotes em *buffer*. Dessa forma, pode-se planejar a disposição de nós e a distribuição de *frames* de acordo com a latência máxima desejada para determinados nós, ou verificar se nós específicos serão capazes de retornar leituras com intervalos apropriados à aplicação.

$$L_{Max}u = T_{slot} \times (d_u + ((|F_{LMT-MAC}| - |F_u|) \times 2) + 1) \quad (3)$$

4. Resultados e Discussão

Nesta seção, os resultados obtidos com as simulações e os testes em plataforma real são apresentados e discutidos. O LMT-MAC foi comparado ao TreeMAC por meio de exaustivas simulações usando o OMNeT++ 4.6 em conjunto com o *framework* INET 3.00. Além disso, foram realizados experimentos com nós reais que comprovam a viabilidade do LMT-MAC. As seguintes topologias foram usadas para avaliação do protocolo:

1. **Linear:** nesta topologia os nós estão organizados em linha e se comunicam apenas com seus vizinhos imediatos. Essa organização está apresentada na Figura 2.

2. **Árvore binária:** a rede foi organizada como uma árvore binária distribuída regularmente pelo espaço. Essa topologia é apresentada na Figura 3.
3. **Disposição dos nós reais:** disposição usada na instalação real da RSSF, permitindo comparar os resultados obtidos na simulação com aqueles obtidos na RSSF real. Além disso, permite verificar o comportamento da RSSF caso o protocolo MAC utilizado fosse o TreeMAC. A topologia é mostrada na Figura 4.

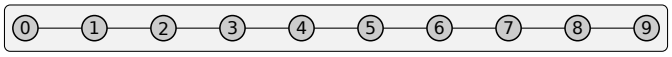


Figura 2. Disposição linear dos nós.

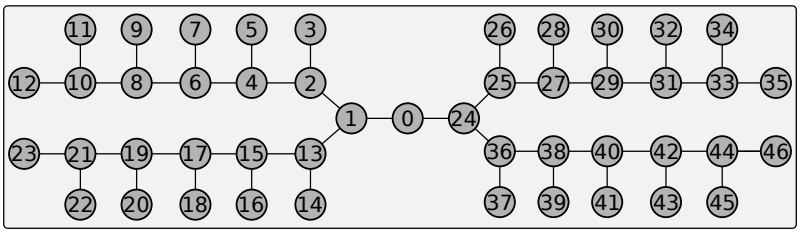


Figura 3. Disposição regular dos nós.

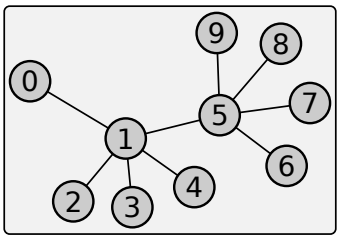


Figura 4. Configuração da rede com a disposição dos nós reais.

As simulações foram feitas com as 3 topologias, enquanto os testes reais foram feitos usando a topologia 3. Os parâmetros das simulações são descritos na Tabela 2. Foram simuladas RSSF com 10, 30 e 47 nós. Cada *slot* tem duração de 20 milissegundos, com tempo de guarda de 1 milissegundo. A taxa de transmissão e a potência foram configuradas para os valores usados nos nós reais, 2 Mbps e 1 mW que são os valores máximos suportados pelo transceptor. Como o protocolo TreeMAC não faz uso de *buffer* para armazenar os pacotes a serem encaminhados, o protocolo foi modificado para usar um *buffer* da mesma forma que o LMT-MAC. Foram usados *buffers* com capacidades para 1, 5, 10 e 20 pacotes.

Um gerador de tráfego constante foi usado para os testes, com configurações de 1, 5, 20 e 40 pacotes por segundo (pps). Cada pacote gerado era armazenado no *buffer* e mantido até que o nó pudesse transmití-lo. Caso a capacidade do *buffer* fosse esgotada, novos pacotes eram descartados e considerados como perdas a serem tratadas pela camada superior.

O tempo de simulação em todos os casos foi de 2000 segundos, com os nós iniciando a transmissão dos pacotes aos 2 segundos para garantir que todos estivessem devidamente configurados, e parando as transmissões aos 1200 segundos, para garantir que todos os pacotes transmitidos pelos nós chegassem ao destino até o final da simulação. Nas simulações foram analisadas, em cada topologia, a taxa de entrega, a vazão e a latência fim-a-fim. Nos testes reais, foram avaliadas a taxa de transmissão ponto-a-ponto e a latência fim-a-fim.

Tabela 2. Parâmetros das simulações.

Parâmetro	Topologia		
	Linear (1)	Árvore (2)	Real (3)
Número de nós	30	47	10
Número de *frames*	29	46	9
Tamanho de *slot*	20 ms		
Tempo de guarda	1 ms		
Taxa de transmissão	2 Mbps		
Potência de transmissão	0 dBm (1 mW)		
Tráfego (pps)	1, 5, 20, 40		
Tamanho de *buffer* (pacotes)	1, 5, 10, 20		
Tempo de simulação	2000 s		

Os testes em plataforma real foram feitos usando 10 nós baseados na plataforma de prototipagem Arduino Uno [Arduino 2015] e no rádio nRF24l01+ [Nordic_Semiconductor 2013], organizados como na topologia mostrada na Figura 4.

Os parâmetros dos testes estão sumarizados na Tabela 3. Foram usados *slots* de 100 milissegundos, tempo de guarda de 20 milissegundos para superar os erros do protocolo de sincronização. Para testes de sincronização essa configuração usou um *frame* adicional. O rádio foi configurado com velocidade de 2 Mbps, potência de 0 dBm (1 mW) e *payloads* de 32 Bytes. O *payload* do quadro transmitido pelo rádio é o espaço reservado para dados das camadas superiores, como os campos de controle usados pelo LMT-MAC. Os nós contavam com espaço em *buffer* para 20 pacotes.

Tabela 3. Parâmetros dos testes reais.

Parâmetro	Valor
Número de nós	10
Número de *frames*	10
Tamanho de *slot*	100 ms
Tempo de guarda	20 ms
Taxa de transmissão	2 Mbps
Potência de transmissão	0 dBm (1 mW)
Tamanho de *payload*	32 Bytes
Tamanho de *buffer*	20 pacotes

Nos testes reais, a taxa de transmissão ponto-a-ponto avaliou a capacidade de comunicação entre dois nós, bem como a quantidade de dados que pode ser trocada em cada *slot*. A latência fim-a-fim foi estimada pela soma dos tempos que os pacotes ficavam armazenados em *buffer* antes de serem encaminhados.

A análise estatística dos resultados foi feita considerando intervalos de confiança de 99% em todos os casos. Nos gráficos de taxa de entrega e vazão os valores para o intervalo de confiança foram omitidos por apresentarem valores demasiadamente pequenos.

4.1. Taxa de Entrega

Para avaliar a taxa de entrega, todos os nós geravam pacotes com destino ao *sink*, e posteriormente foi verificado quantos pacotes foram efetivamente entregues ao destino. Da diferença entre os dois valores obteve-se a quantidade de pacotes perdidos. Com isso, é possível avaliar a confiabilidade do protocolo MAC em RSSF com determinados tipos de tráfego e com diferentes tamanhos de *buffer*.

Os gráficos das Figuras 5, 6 e 7, apresentam a taxa de entrega obtida respectivamente para a topologia linear, árvore binária e disposição dos nós reais. Os resultados

mostram que a taxa de entrega dos protocolos depende diretamente das capacidades dos *buffers* utilizados. Quando os *buffers* são suficientemente grandes para a quantidade de tráfego gerado, ou a rede possui poucos nós, ambos protocolos conseguem entregar 100% dos pacotes.

Com o aumento da quantidade de nós, o tamanho de ciclo também cresce, o que faz com que os nós tenham oportunidades de transmissão em intervalos maiores. Assim, com o gerador de tráfego criando novos pacotes constantemente, e os nós tendo menos chances de transmití-los, a capacidade dos *buffers* é esgotada, e mais pacotes são descartados, como notado nos gráficos das Figuras 5 e 6. Com um ciclo menor e mais comunicações simultâneas, o LMT-MAC permite que os pacotes fiquem retidos em *buffer* por menos tempo que no TreeMAC, liberando espaço para os novos pacotes gerados e reduzindo as perdas por escassez de *buffer*.

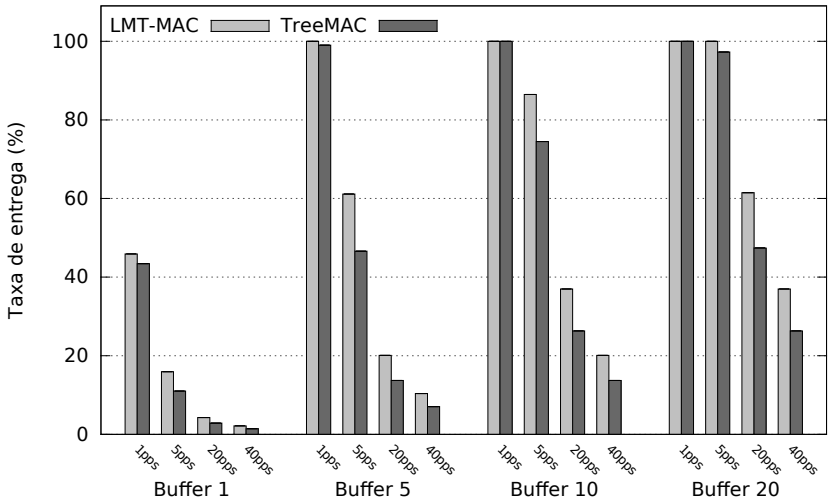


Figura 5. Taxa de entrega para a topologia linear.

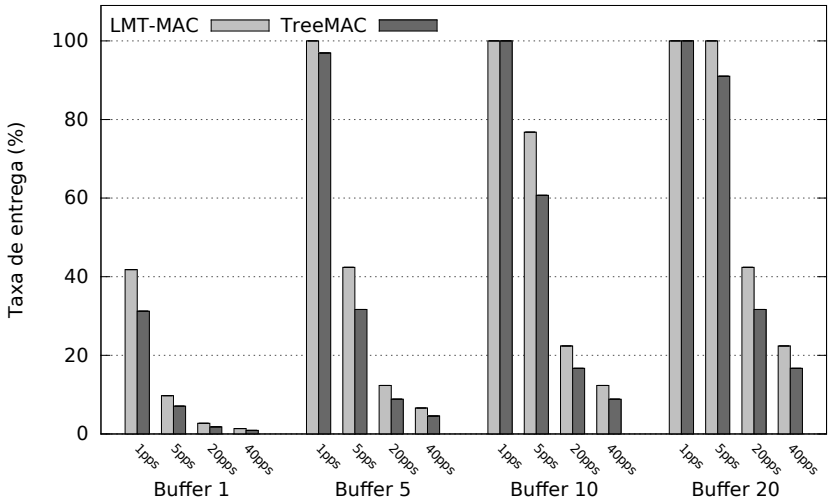


Figura 6. Taxa de entrega para árvore binária.

4.2. Vazão

A avaliação da vazão se deu pela análise da quantidade de pacotes por segundo que chegavam ao *sink*. Nas simulações foram usados pacotes de 321 bits, mesmo tamanho dos

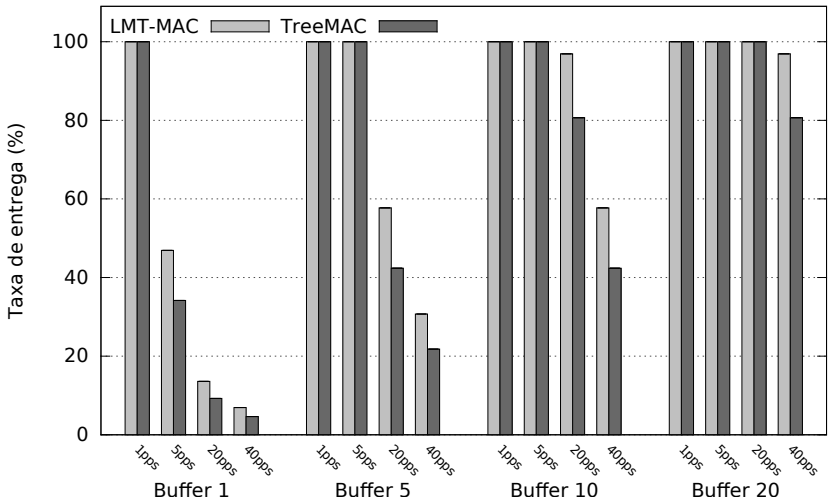


Figura 7. Taxa de entrega para a disposição real.

pacotes gerados pelo transceptor utilizado nos testes reais. Os gráficos das Figuras 8, 9 e 10 trazem os resultados obtidos com os testes de vazão.

A análise dos gráficos mostra que o LMT-MAC apresenta vazão superior ao TreeMAC na maioria dos casos. Quando a quantidade de tráfego gerada é baixa, ambos protocolos apresentam o mesmo comportamento. O LMT-MAC atinge maior vazão quando há mais tráfego sendo gerado, por permitir mais comunicações simultâneas devido ao uso de multicanal. Além disso, a divisão de cada *frame* em apenas dois *slots* permite que as mensagens cheguem ao *sink* em intervalos menores, resultando em maior vazão.

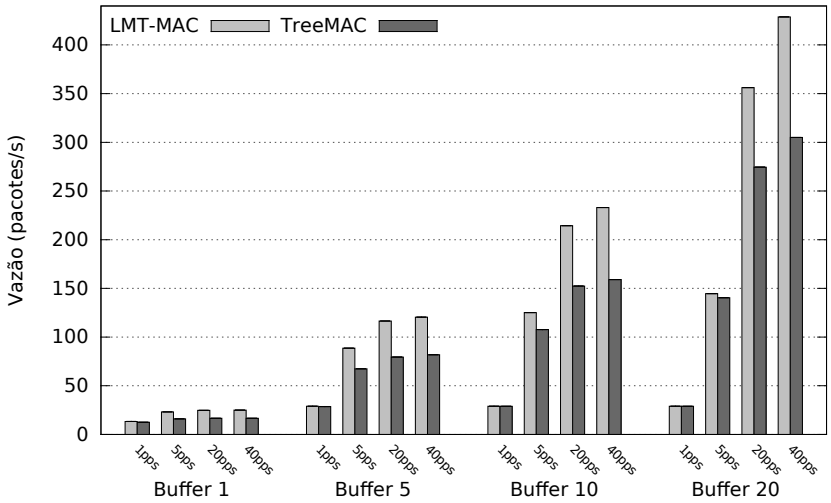


Figura 8. Vazão de dados para o *sink* na topologia linear.

4.3. Latência

Para a avaliação da latência, mediu-se o tempo levado desde a criação de um pacote em um nó, até o seu processamento no nó de destino. Na disposição linear, cujo gráfico é apresentado na Figura 11, pode-se notar o impacto da distância no crescimento da latência nos dois protocolos. Novamente, o LMT-MAC apresenta resultados superiores por ser

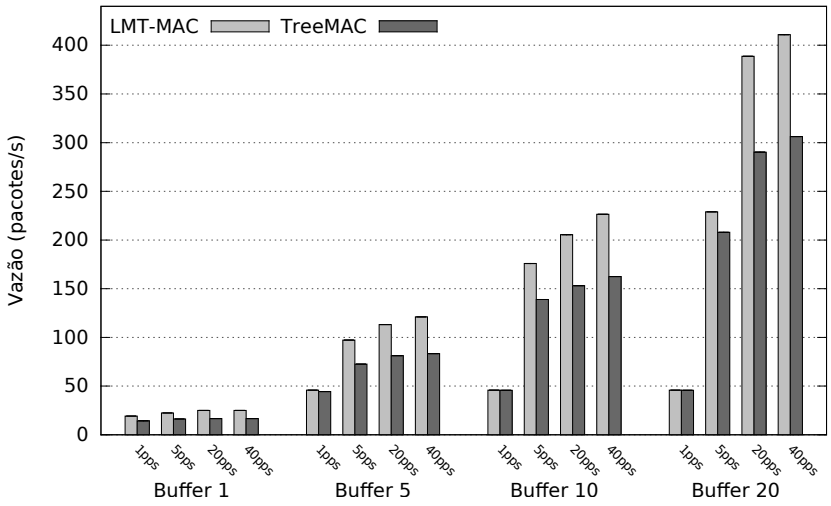


Figura 9. Vazão de dados para o *sink* na árvore binária.

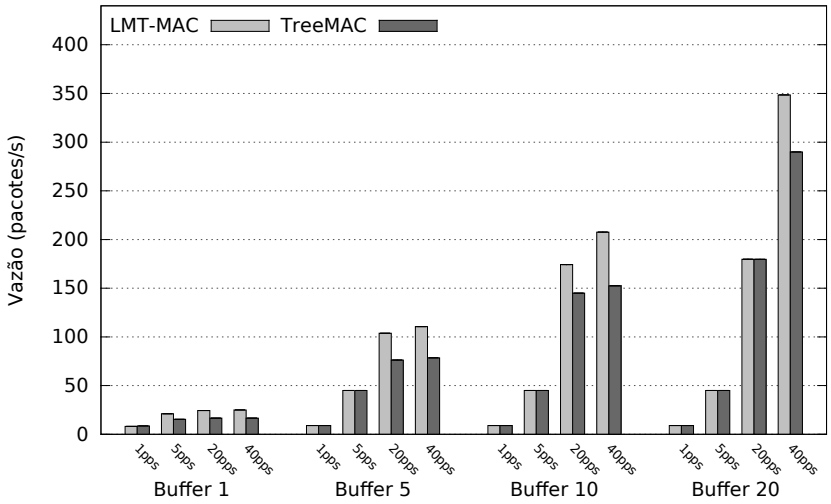


Figura 10. Vazão de dados para o *sink* disposição real.

multicanal e usar menos *slots* por *frame*. Com *frames* menores, os pacotes ficam retidos em *buffer* por menos tempo, dado que os intervalos de transmissão são menores e a quantidade de comunicações simultâneas é maior, devido ao uso de multicanal.

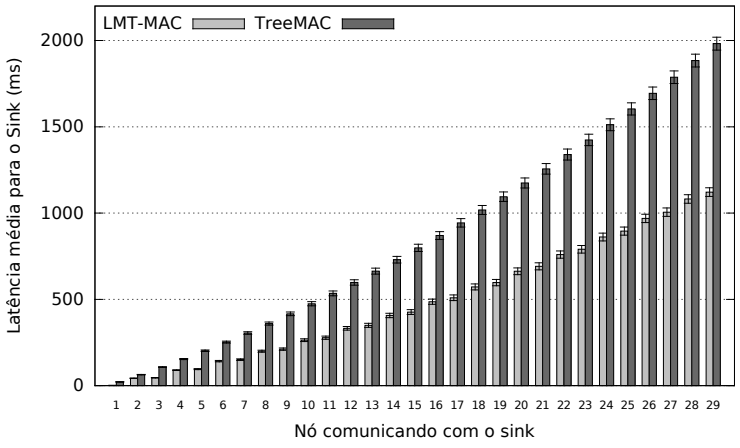


Figura 11. Latência para o *sink* na topologia linear.

O gráfico da Figura 12 apresenta os resultados obtidos para a disposição em árvore binária. Para apresentação dos dados, foi selecionada uma sub-árvore contendo os nós de 1 a 12, e os resultados obtidos para as demais sub-árvores da rede apresentaram o mesmo comportamento. O LMT-MAC obteve latência inferior ao TreeMAC em todos os casos. Em ambos protocolos, os nós que têm mais filhos, e, portanto, têm mais *frames*, apresentam menor latência. No entanto, no LMT-MAC o crescimento da latência é menor, devido à quantidade menor de *slots* por *frame*.

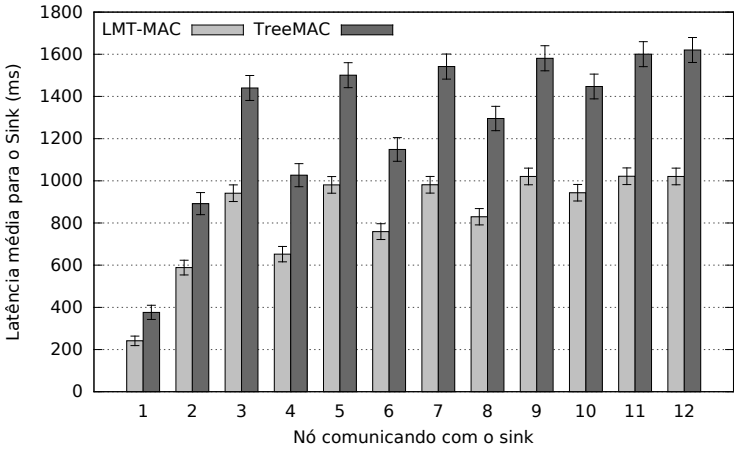


Figura 12. Latência para o *sink* na árvore binária.

No gráfico da Figura 13 são apresentados os resultados para a disposição dos nós reais. A semelhança dos resultados obtidos para nós que possuem a mesma configuração (distância para o *sink* e quantidade de *frames*) em ambos protocolos mostra a possibilidade de predição de tais valores, como é o caso dos nós 2 a 4, e dos nós 6 a 9. O nó 5, por sua vez, está à mesma distância do *sink* que os nós 2 a 4, porém, apresenta latência significativamente inferior por ter mais *frames* de transmissão. Em todos os casos, o LMT-MAC apresentou latência consideravelmente inferior ao TreeMAC, da mesma forma que nos casos anteriores, devido à menor quantidade de *slots* por *frame* e uso de multicanal.

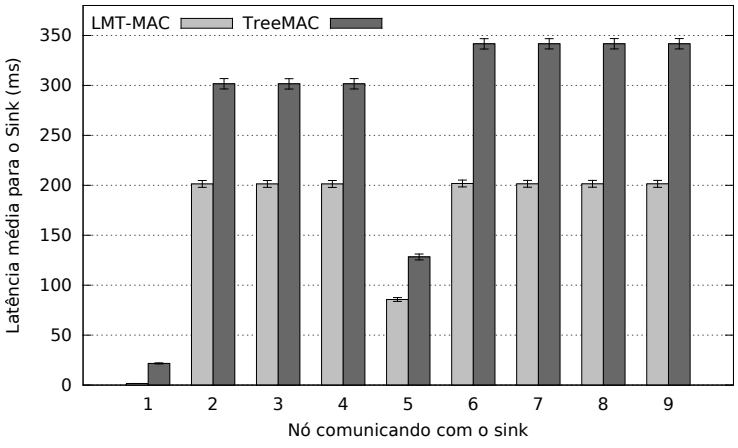


Figura 13. Latência para o *sink* na disposição real.

Nos testes com nós reais, os valores médios obtidos para latência são apresentados na Tabela 4. A tabela traz os tempos mínimo, máximo e médio para transmissão dos pacotes para o *sink* a partir de nós em posições variadas na rede. Pode-se notar que

mesmo estando à mesma distância (em saltos) para o *sink*, a comunicação com o nó 5 tem latência consideravelmente inferior em relação ao nó 2, de modo semelhante ao obtido nas simulações, devido à maior quantidade de *frames*.

Tabela 4. Latência de transmissão para o *sink*.

Origem	Saltos	Tempo mínimo	Tempo máximo	Tempo médio
Nó 1	1	1,66 ms	397,59 ms	115,33 ms
Nó 2	2	115,50 ms	4.097,30 ms	2.091,11 ms
Nó 5	2	103,54 ms	2.495,06 ms	920,86 ms
Nó 6	3	312,90 ms	8.221,24 ms	4.362,53 ms

4.4. Taxa de Transmissão

Nos testes com nós reais, a taxa de transmissão foi testada com o envio de 20 pacotes (capacidade do *buffer*) entre pares de nós. Cada pacote tem um tamanho total de 321 bits, e foram transmitidos em aproximadamente 23 milissegundos, tanto em *upstream* (do nó filho para o nó pai), quanto em *downstream* (do nó pai para o nó filho), obtendo uma vazão média de aproximadamente 270 Kbps, transmitindo um total de 802,5 Bytes. A Tabela 5 sumariza os resultados obtidos.

Tabela 5. Taxa de transmissão entre dois nós.

	Nó filho $\rightarrow$ Nó pai	Nó pai $\rightarrow$ Nó filho
Pacotes enviados/frame	20	20
Tempo médio de transmissão	$23,72\ ms\ \ (\pm\ 0,043\ ms)$	$23,65\ ms\ \ (\pm\ 0,005\ ms)$
Velocidade	$270.608\ bps\ \ (\pm\ 490\ bps)$	$271.411\ bps\ \ (\pm\ 59\ bps)$
Total transmitido/frame	$802,5\ B$	$802,5\ B$

5. Conclusão

Os resultados obtidos nas simulações mostram que com o uso de multicanal, e consequente redução da quantidade de *slots* necessários em um *frame* para evitar colisões, fez com que o LMT-MAC obtivesse resultados superiores ao TreeMAC na vasta maioria dos casos. A necessidade dos nós reterem pacotes em *buffer* até que tenham uma oportunidade de transmissão, faz com que a taxa de entrega dos protocolos dependa diretamente do tamanho do ciclo, do tamanho do *buffer* e da quantidade de tráfego gerada. Por usar um *slot* a menos que o TreeMAC, e empregar multicanal, o LMT-MAC consegue entregar mais pacotes e com menor latência. Pelos mesmos motivos, a capacidade de vazão obtida com o LMT-MAC é superior.

A implementação em nós reais implicou em modificação nos tempos de operação, devido a limitações de hardware dos nós sensores (32 KB de memória de programa e 2 KB de memória RAM), exigindo um método mais simples e menos eficaz de sincronização de tempo. Isso resultou na definição de um *slot* maior (100 ms em nós reais contra 20 ms nas simulações), e em tempos de guarda mais altos (20 ms nos nós reais e 1 ms nas simulações) para superar os erros de sincronização. No entanto, essa implementação foi indispensável para avaliar o funcionamento do protocolo em plataformas reais, e criar uma base para sua aplicação em outros nós sensores mais robustos. O LMT-MAC se mostrou um protocolo previsível, apresentando latência média dentro dos limites calculados analiticamente, e obtendo taxas de vazão e entrega superiores ao TreeMAC.

Como trabalhos futuros, pretende-se desenvolver um mecanismo adaptativo para uso dos *slots*, permitindo reduzir a latência, e implementar o protocolo modificado em uma rede real composta por mais nós.

6. Agradecimentos

Os autores agradecem o apoio financeiro das agências CAPES, CNPq e FAPEMIG.

Referências

Abdeli, D., Zelit, S., and Moussaoui, S. (2013). RTH-MAC: A real time hybrid MAC protocol for WSN. In *Programming and Systems (ISPS), 2013 11th International Symposium on*, pages 153–162.

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):393 – 422.

Arduino (2015). Arduino. `http://arduino.cc/`. Acessado em 01-06-2015.

Gungor, V. and Hancke, G. (2009). Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE Transactions on*, 56(10):4258–4265.

HART (2015). Hart Communication Foundation. `http://www.hartcomm.org/`. Acessado em 01-06-2015.

Incel, O. (2011). A survey on multi-channel communication in wireless sensor networks. *Computer Networks*, 55(13):3081–3099. cited By 28.

Incel, O. D., van Hoesel, L., Jansen, P., and Havinga, P. (2011). MC-LMAC: A multi-channel MAC protocol for wireless sensor networks. *Ad Hoc Networks*, 9(1):73 – 94.

Instruments, T. (2006). CC2420: 2.4 GHz IEEE 802.15. 4/ZigBee-ready RF Transceiver. *Available at Available at http://www. ti. com/lit/gpn/cc2420*, page 53.

Nordic_Semiconductor (2013). *nRF24l01 Product Specification v2.0*.

Romer, K. and Mattern, F. (2004). The design space of wireless sensor networks. *Wireless Communications, IEEE*, 11(6):54–61.

Shen, W., Zhang, T., Barac, F., and Gidlund, M. (2014). PriorityMAC: A Priority-Enhanced MAC Protocol for Critical Traffic in Industrial Wireless Sensor and Actuator Networks. *Industrial Informatics, IEEE Transactions on*, 10(1):824–835.

Song, W.-Z., Huang, R., Shirazi, B., and LaHusen, R. (2009). TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks. *Pervasive and Mobile Computing*, 5(6):750 – 765. PerCom 2009.

Trilha Principal do SBRC 2016
Sessão Técnica 18
Redes Ópticas

Uma Estratégia Híbrida e Iterativa para Roteamento e Posicionamento de Nós OEO em Redes Ópticas Translúcidas

Gilvan Durães[1], Víctor Araújo[2], André Soares[3], José Monteiro[4], William Giozza[5]

[1]Instituto Federal de Educação, Ciência e Tecnologia Baiano (IF Baiano) – *Campus* Catu –, BA – Brasil

[2]QUALCOMM Global Trading Pte. Ltd. – QGT PMIC Singapore – No. 6 Serangoon North Ave 5, 554910 – Singapore

[3]Programa de Pós-Graduação em Ciência da Computação – Universidade Federal do Piauí (UFPI) – Teresina, PI – Brasil

[4]Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE) – Recife, PE – Brasil

[5]Departamento de Engenharia Elétrica – Universidade de Brasília (UnB) – Brasília, DF – Brasil

gilvan.duraes@catu.ifbaiano.edu.br, giozza@unb.br

Resumo. Este trabalho propõe uma solução integrada para os problemas de dimensionamento e posicionamento de nós regeneradores OEO e roteamento multirrestritivo em redes ópticas translúcidas cientes das limitações da camada física óptica. A nova estratégia proposta, chamada *Iterative Approach for Routing and Regenerator Placement* (IARRP) é comparada com outros algoritmos de dimensionamento e posicionamento de nós OEO sob as métricas de utilização e dimensionamento de nós OEO. Além disso, por ser também, simultaneamente, uma estratégia de roteamento multirrestritivo, a IARRP é comparada com outras duas estratégias de roteamento multirrestritivo sob diferentes métricas de desempenho e topologias de rede. A estratégia proposta IARRP apresentou os melhores resultados em todos os cenários avaliados.

Abstract. *This work proposes a new integrated solution for the OEO regenerator nodes placement and multirrestrictive routing problems for physical layer impairment aware translucent optical networks. The new strategy named Iterative Approach for Routing and Regenerator Placement (IARRP) is compared in terms of OEO utilization and number of OEO nodes required in the networks with other OEO nodes placement algorithms proposed in the literature. Furthermore, the IARRP approach is compared with other multirrestricted routing algorithms. The evaluated scenarios include several representative topologies for translucent optical networking. For all studied scenarios, IARRP achieved superior performance.*

1. Introdução

A tecnologia de multiplexação por divisão de comprimento de onda (*Wavelength Division Multiplexing* – WDM), a qual permite dividir a banda passante de uma fibra óptica em diversos canais ópticos de comunicação, é uma abordagem viável para

melhor utilizar a largura de banda da fibra óptica e atender ao rápido crescimento do tráfego da Internet [1][2].

Uma nova arquitetura de rede óptica WDM que utiliza conversor *Óptico-Eletro-Óptico* (OEO) em alguns nós intermediários da rede e comutador puramente óptico em outros nós, se torna cada vez mais uma realidade. Essa arquitetura de rede óptica que tem como objetivo agregar a agilidade de uma rede óptica transparente à qualidade do sinal óptico garantido com a conversão OEO é conhecida como rede óptica translúcida [3][5][4]. Os nós que utilizam a conversão OEO para regenerar o sinal óptico são conhecidos por nós regeneradores OEO ou, simplesmente, nós OEO [3].

Para o estabelecimento de um circuito óptico (*lightpath*) entre um par de nós (origem, destino) de uma rede óptica, denotado neste trabalho por *par(o,d),* é necessário definir uma rota e alocar um comprimento de onda em cada enlace dessa rota. O problema de roteamento e alocação de comprimento de onda, visando atender um maior número de solicitações dinâmicas de *lightpath* é conhecido como *Routing and Wavelength Assignment* (RWA) ou RWA dinâmico [2], [3], [6], [7].

Em [7] é apresentado um dos primeiros estudos de análise de desempenho dos algoritmos RWA em redes ópticas modeladas com degradações na camada física. Além de avaliar a disponibilidade de rotas e comprimentos de onda, os autores evidenciam a necessidade dos algoritmos de RWA considerarem as degradações de camada física para a escolha de recursos apropriados para o estabelecimento de circuitos ópticos.

Para incluir esta limitação não tratada nos algoritmos RWA pioneiros, foram concebidos os algoritmos da classe *Physical Layer Impairments constrained and aware RWA* (PLI-RWA) [7], [8], [9]. Os algoritmos PLI-RWA são algoritmos de roteamento e alocação de comprimento de onda cientes das restrições associadas à degradação da camada física óptica. O objetivo de um algoritmo PLI-RWA consiste em encontrar uma rota e um comprimento de onda para o estabelecimento de um circuito óptico, levando em consideração as degradações da camada física óptica (*e.g.,* atenuação da fibra, perdas de inserção, ruído da emissão espontânea de amplificador óptico, dispersão por modo de polarização e *crosstalk*) [8]. Desta forma, o problema de roteamento dinâmico ciente das degradações da camada física em redes ópticas translúcidas é considerado mais difícil que o problema correspondente para redes ópticas transparentes [2][7].

Com o objetivo de atender a uma lista de restrições ou requisitos impostos aos algoritmos PLI-RWA, são concebidos modelos e algoritmos de roteamento multirrestritivos [10][11]. Este trabalho utiliza a abordagem proposta em [11] para a modelagem de múltiplas restrições associadas às degradações da camada física óptica. Nesta abordagem, as restrições podem estar associadas ao comprimento de onda (*e.g. link crosstalk*), ao enlace óptico (*e.g.* distância) ou ao nó óptico (*e.g. switch crosstalk*). Neste caso, uma Rota Candidata Viável (RCV) é uma rota que atende a todas as restrições impostas ao roteamento.

Neste trabalho é adotado o tipo de rede óptica translúcida com posicionamento esparso de nós regeneradores OEO. Neste tipo de rede óptica translúcida, os nós OEO devem ser estrategicamente posicionados na rede de modo a apresentarem o melhor rendimento em termos de regeneração do sinal óptico e ganho na probabilidade de bloqueio de requisições ópticas. Além disso, outro objetivo das estratégias de posicionamento de nós OEO é minimizar a quantidade necessária de nós regeneradores OEO na rede óptica translúcida, uma vez que, apesar de os nós OEO proporcionarem

uma melhor qualidade do sinal óptico, eles aumentam o custo de projeto da rede e inserem atrasos na comunicação óptica [5], [4].

Neste contexto, este trabalho propõe um algoritmo genético para roteamento fixo e, além disso, propõe uma nova estratégia híbrida que soluciona, simultaneamente, o problema de posicionamento esparso de nós regeneradores e o problema de roteamento multirrestritivo em redes ópticas translúcidas. Os resultados de simulação mostram o melhor desempenho da estratégia proposta quando comparada com outros algoritmos de posicionamento esparso de nós regeneradores e algoritmos de roteamento multirrestritivo.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 são apresentados os trabalhos relacionados; na Seção 3 é apresentado o algoritmo genético GASTL que foi proposto e utilizado como módulo na proposta principal; na Seção 4 é apresentada a estratégia proposta *Iterative Approach for Routing and Regenerator Placement* (IARRP); na Seção 5 são discutidos os resultados de avaliação de desempenho da estratégia proposta IARRP comparando-a com outros algoritmos propostos na literatura; por fim, na Seção 6 são apresentadas as considerações finais.

2. Trabalhos Relacionados

O problema de dimensionamento e posicionamento de nós regeneradores OEO é uma especificidade do *Facility Location Problem* [16], sendo um problema de combinatória. Desta forma, a exploração de todas as combinações é computacionalmente inviável, tornando as propostas de heurísticas de dimensionamento e posicionamento plenamente justificadas [11], [13], [4], [14], [15], [17].

Uma heurística simples, por exemplo, seria realizar o roteamento de menor caminho e posicionar os regeneradores OEO nos nós que apresentarem maior número de *lightpaths* transitando, ou seja, nós que mais vezes são nós intermediários de *lightpaths*. Na literatura é possível encontrar diversas estratégias baseadas na predição da carga de tráfego da rede e/ou em simulações prévias e/ou programação linear, tais como *Transitional Weight* [11], *Genetic Aproach [15], Iterative Regenerator Placement* [17]. *Traffic Load Prediction* [18], *Signal Quality Prediction* [18], *Most Used Regenerator Placement e Simultaneous Used Regenerator Placement* [19], A maioria destes trabalhos tem como foco o problema IA-RWA estático, onde os *lightpaths* são previamente conhecidos.

Considerando apenas a topologia da rede como entrada, sem a exigência de simulações prévias nem de predição da carga de tráfego e, além disso, considerando o cenário de tráfego dinâmico, as seguintes heurísticas de posicionamento de nós OEO foram propostas na literatura: *Centered Node First, Random Placement Algorithm, Nodal Degree First* [18], e *Hub Node First* [13]. Segundo os estudos realizados em [18], o algoritmo *Nodal Degree First* (NDF) apresentou melhores resultados quando comparado com as estratégias *Centered Node First* e *Random Placement Algorithm*, em todos cenários avaliados. Adicionalmente, o algoritmo *Hub Node First* (HNF) é proposto em [13] e citado em [5] e [4] como sendo um algoritmo avançado para posicionamento de nós regeneradores baseado em informações topológicas. Desta forma, os algoritmos NDF e HNF foram utilizados neste trabalho.

O algoritmo NDF, proposto em [18] prioriza os nós que apresentam maior grau físico (número de interfaces físicas) no posicionamento dos nós OEO. Os nós são

distribuídos até que todos os pares de nós (origem, destino) sejam alcançáveis. Já o algoritmo HNF [13] posiciona nós OEO priorizando os nós que possuem maior grau lógico. Um grafo lógico é criado com todos nós da topologia. Nesse grafo existe um enlace lógico entre cada par de nós, caso esses nós possam alcançar um ao outro. Desta forma, o grau lógico de um nó da rede óptica é o número de nós destinos que são alcançáveis por esse nó. O algoritmo finaliza quando todos nós da rede alcançarem todos os outros demais nós da rede [13]. Uma extensão do algoritmo HNF foi proposto em [14] para suportar cenários de sobrevivência a falhas.

Em [11] foram propostos os algoritmos *Offline Multi-Restrictive Routing* (Off-MRR) e *Multi Restrictive Routing* (MRR) para atender à nova modelagem de redes ópticas translúcidas com múltiplas restrições. Os algoritmos Off-MRR e MRR foram comparados em termos de probabilidade de bloqueio, utilização da rede e tempo de execução. O algoritmo Off-MRR é uma estratégia de roteamento fixo que leva em consideração todas as restrições do tipo *offline*, apenas. Já o MRR, além de pré-computar as rotas que atendem às restrições do tipo *offline*, retorna, durante a operação da rede, a primeira rota viável encontrada, considerando todas as restrições impostas à rede. Os algoritmos Off-MRR e MRR são utilizados neste trabalho.

3. Roteamento Fixo com Algoritmo Genético

Esta seção apresenta o algoritmo genético proposto *Genetic Algorithm for Shortest Translucent Lightpath* (GASTL) que é utilizado como módulo na estratégia principal proposta e apresentada na Seção 4.

O algoritmo genético é um método heurístico proposto por John Holland em 1975, inspirado na teoria evolutiva [20]. Os algoritmos genéticos possuem diversos conceitos presentes na teoria biológica como a mutação, cruzamento e seleção natural. No algoritmo genético uma população inicial é criada através do sorteio de suas características de forma aleatória. A cada geração é calculada a adaptabilidade através de uma função de avaliação para cada indivíduo da população. Dois pais se cruzam para formar dois filhos, os quais podem sofrer mutação. O processo de cálculo da adaptabilidade e reprodução é repetido até se alcançar a última geração.

A seguir são apresentadas as principais etapas do algoritmo genético.

Etapas do Algoritmo Genético
1) Gera população inicial aleatoriamente;
2) Calcula adaptabilidade;
3) Ordena os indivíduos de acordo à adaptabilidade de cada um deles;
4) Realiza o cruzamento e a mutação;
5) Verifica se alcançou a última geração. Se sim, finaliza o algoritmo. Se não, retorna para o Passo 2.

3.1. Modelagem do Cromossomo Base

No cromossomo base, cada gene representa uma rota viável de menor caminho para um determinado par de nós (origem, destino). Vale ressaltar que este mapeamento é realizado apenas para os pares de nós (origem, destino) que apresentam mais de uma Rota Candidata Viável (RCV). A Figura 1 exemplifica a modelagem do cromossomo base para uma topologia pequena com restrição de 2 saltos. Os pares de nós que apresentam apenas um menor caminho viável (RCV – 1 = 0) são descartados da

representação do cromossomo. Tal conduta não apenas reduz a utilização da memória como simplifica a modelagem do cromossomo do algoritmo genético.

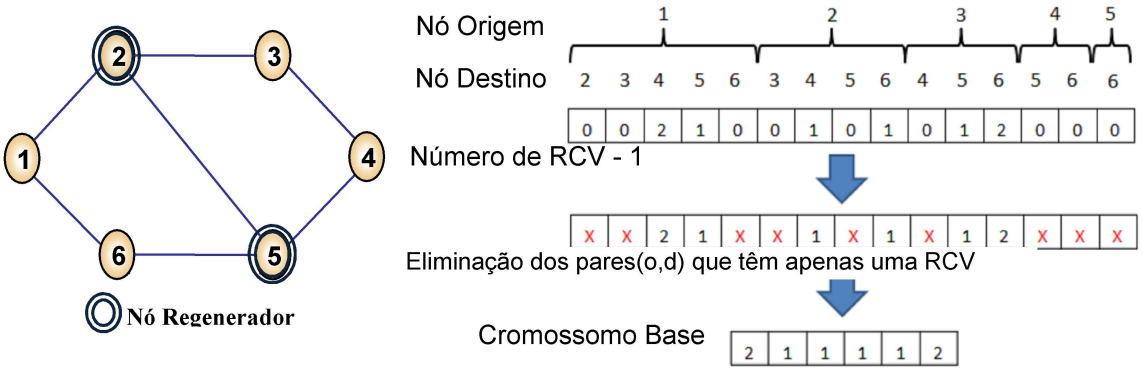


Figura 1. Formação do cromossomo base

A partir do cromossomo base é gerada a população inicial através do sorteio de números naturais entre zero e o valor de cada gene do cromossomo base. Após a geração dos primeiros indivíduos, a cada próxima geração os indivíduos têm a sua adaptabilidade avaliada por uma função objetivo. Neste algoritmo, os indivíduos mais aptos são os que resultam em uma configuração de rotas que possuem os enlaces menos carregados em termos de número de rotas por enlace.

3.2. Seleção

Os cromossomos pais são selecionados através do método da roleta [20]. Nesta fase é gerado um número aleatório entre zero e a soma da adaptabilidade de todos os indivíduos. Desta forma, os indivíduos são percorridos em ordem e sua adaptabilidade é somada, quando o valor da soma alcança ultrapassa o valor sorteado, o indivíduo é selecionado. No método da roleta, a probabilidade de o indivíduo ser sorteado é igual ao valor percentual da sua adaptabilidade na soma de todos os indivíduos. Assim, indivíduos com melhor adaptabilidade apresentam maior probabilidade de serem escolhidos pela roleta. A adaptabilidade de um indivíduo é dada pela função de avaliação do indivíduo.

A função de avaliação (também chamada de *fitness*) deve ser configurada para qualificar quão bom é um determinado cromossomo (representando, no caso, uma solução candidata de roteamento). Neste sentido, foram utilizadas funções baseadas na frequência de uso dos enlaces. Entende-se por frequência de uso do enlace o número de rotas que passam em um determinado enlace da rede. Duas funções de avaliação foram propostas e comparadas, são elas: Menor Pico (MP) e Menor Desvio Padrão (MDP). A primeira função (MP) é inversamente proporcional ao número de rotas que passam no enlace mais sobrecarregado da rede. A segunda (MDP) é inversamente proporcional ao desvio padrão da frequência de uso dos enlaces da rede.

3.3. Cruzamento e Mutação

A etapa de cruzamento (ou *crossover*) é realizada em *n* pontos do cromossomo, onde *n* corresponde a um valor gerado aleatoriamente que vai de zero ao número de genes menos um. Uma vez definidos os pontos de *crossover*, seus genes são trocados, formando dois cromossomos filhos recombinados. Com o objetivo de evitar um mínimo local é utilizada a técnica de mutação. A mutação ocorre após a operação de *crossover* e

possui baixa probabilidade de ocorrência. Na ocorrência de mutação, é alterado o valor de um gene aleatoriamente. A Figura 2 ilustra o procedimento de geração de filhos.

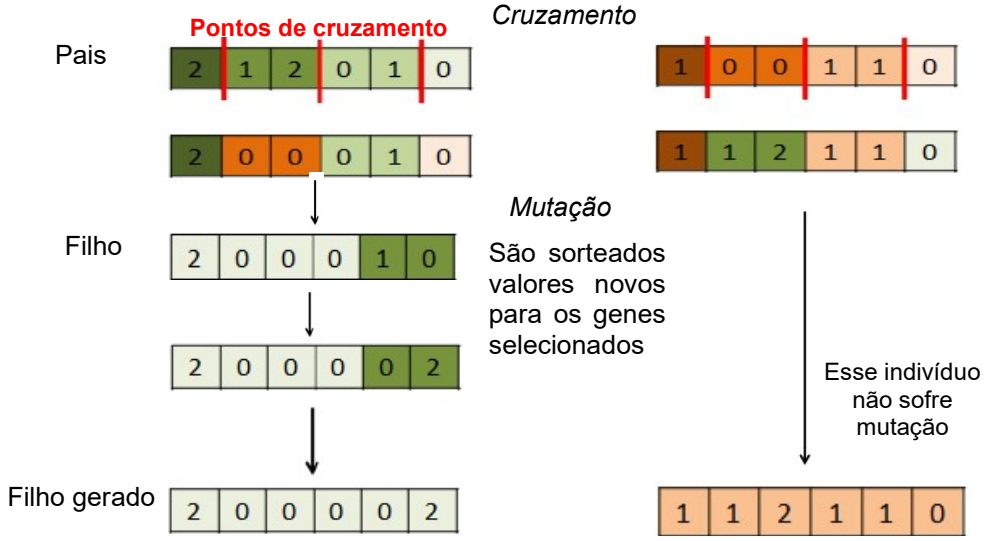


Figura 2. Cruzamento e mutação na geração de filhos.

Com o objetivo de evitar que o melhor indivíduo seja perdido, por não ser mantido no procedimento de seleção ou por sofrer uma mutação, o conceito de *elitismo* [20] é aplicado para obter sempre soluções melhores ou iguais às soluções obtidas nas gerações anteriores. A taxa de elitismo, dada como entrada para o algoritmo, representa o percentual de melhores indivíduos de uma geração que devem permanecer para a geração seguinte.

4. Estratégia híbrida e iterativa para o roteamento multirrestritivo e posicionamento de nós regeneradores OEO

Esta seção apresenta a estratégia proposta principal que tem como objetivos dimensionar e posicionar regeneradores OEO, assim como, simultaneamente, realizar o roteamento multirrestritivo em redes ópticas translúcidas.

Esta nova estratégia chamada de *Iterative Approach for Routing and Regenerator Placement* (IARRP) apresenta duas etapas. A primeira etapa define as possíveis soluções de posicionamento de nós regeneradores OEO. Esta etapa é inspirada no algoritmo HNF apresentado na Seção 2. Semelhantemente ao HNF, esta primeira etapa da estratégia IARRP se baseia no conceito de maior grau lógico para a escolha de um nó regenerador OEO. No entanto, ele utiliza a técnica de *backtracking* para escolher um "segundo caminho" durante a iteratividade do algoritmo. Na segunda etapa, a estratégia proposta IARRP utiliza como módulo o algoritmo genético GASTL (Seção 3) para auxiliar na escolha da melhor solução de posicionamento de nós OEO, bem como, para definição do roteamento. Nesta segunda etapa, a estratégia IARRP avalia as soluções parciais de posicionamento de nós regeneradores fornecidas na etapa anterior, por meio do módulo do algoritmo genético GASTL que realiza o roteamento fixo (*offline*) multirrestritivo ciente das degradações da camada física óptica. Desta forma, além do posicionamento de nós OEO, outra saída da estratégia proposta IARRP é a solução de rotas encontradas pelo algoritmo GASTL.

A seguir são apresentadas as principais etapas da estratégia proposta IARRP.

Etapas da estratégia IARRP:

[Etapa 1] – Nesta etapa são determinadas as possíveis soluções de posicionamento de nós regeneradores.

1) Calcular o grau lógico de cada nó da rede óptica, ou seja, o número de nós destinos que são alcançáveis por cada nó.

2) Posicionar regenerador OEO no nó que apresenta maior grau lógico. No caso de mais de um nó apresentar o maior grau lógico, utilizar o critério de maior grau físico (maior número de interfaces) como critério de desempate.

3) Utilizar a técnica de *backtracking* para escolher o segundo nó apto a ser nó OEO e continuar outra linha de execução do algoritmo para a segunda opção de nó OEO.

4) Verificar se todos os pares de nós (*o,d*) são alcançáveis. Caso negativo, retornar ao passo 1 para posicionar novo nó OEO. Caso positivo, seguir para a Etapa 2.

[Etapa 2] – Nesta etapa são conhecidas todas as possíveis soluções de posicionamento de nós OEO encontradas na etapa anterior.

5) Para cada solução de posicionamento de nós regeneradores P_i faça:

5.a Utilizando o algoritmo de *Dijkstra* modificado [11], calcular o conjunto *R* de todas as possíveis rotas simples para cada *par(o,d)*.

5.b Remover de cada conjunto *R* aquelas rotas que não satisfazem às restrições *offline* (ou por enlace, ou por comprimento de onda, ou por nó óptico).

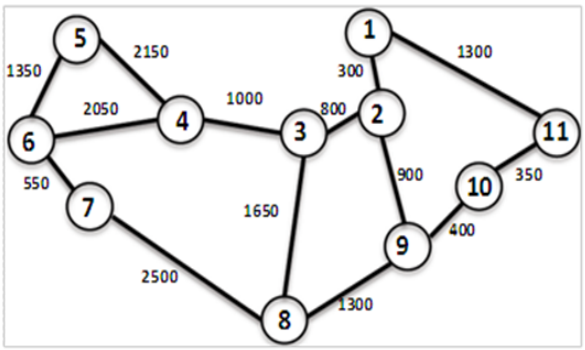
5.c Manter no conjunto *R* apenas as rotas de menor caminho em termos de número de saltos.

5.d Executar o algoritmo GASTL, tendo como entrada as rotas candidatas do conjunto *R*.

5.e Guardar a solução de rotas R_i encontrada pelo algoritmo GASTL. O valor da função de avaliação para P_i será o mesmo valor da função de avaliação para R_i.

6) Retornar as soluções P_i e R_i que obtiveram melhor avaliação.

A Figura 3 ilustra o passo-a-passo do algoritmo IARRP para a topologia da rede Abilene, considerando as restrições *offline* de 2 saltos e distância (2500Km).

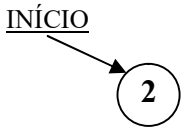
Passo 1 – Cálculo do grau lógico de cada nó da rede óptica	
Nó 1, grau lógico: 5 *Nó 2, grau lógico: 7* *Nó 3, grau lógico: 5* *Nó 4, grau lógico: 4* *Nó 5, grau lógico: 3* *Nó 6, grau lógico: 3* *Nó 7, grau lógico: 3* *Nó 8, grau lógico: 5* *Nó 9, grau lógico: 6* *Nó 10, grau lógico: 5* *Nó 11, grau lógico: 4*	

Passo 2 – Posicionamento de nó OEO

Melhor nó: 2

Grau lógico: 7

Sequência do posicionamento de nós:

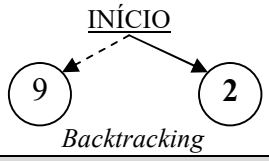


Passo 3 –Uso da técnica de backtracking

Segundo Melhor nó: 9

Grau lógico: 6
Grau físico:3

Sequência do posicionamento de nós:

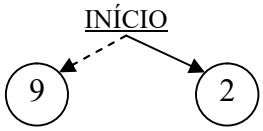


Passo 4 – Verificação da alcançabilidade dos pares de nós

Linha de execução1:
Nó posicionado: 2.
Todos os pares de nós (*o,d*) são alcançáveis? Não.

Linha de execução2:
Nó posicionado: 9.
Todos os pares de nós (*o,d*) são alcançáveis? Não.

Sequência do posicionamento de nós:



Execução dos Passos de 1 a 4 iterativamente...

Última execução do Passo 4 – Verificação da alcançabilidade dos pares de nós

Linha de execução 1:
-Nós OEO posicionados: **2, 4, 6**.
- Todos os pares de nós (*o,d*) são alcançáveis? **Sim**.

Linha de execução 2:
- Nós OEO posicionados: 9, 2, 4
- Todos os pares de nós (*o,d*) são alcançáveis? Não.

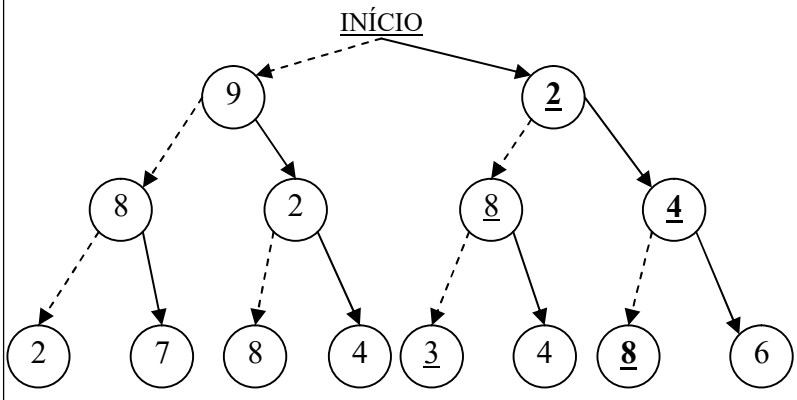
Linha de execução 3:
- Nós OEO posicionados: **2, 8, 4**.
- Todos os pares de nós (*o,d*) são alcançáveis? **Sim**.

Linha de execução 4:
-Nós OEO posicionados: 2, 4, 8.
solução repetida!

Linha de execução 5:
- Nós OEO posicionados: 9, 8, 7.
- Todos os pares de nós (*o,d*) são alcançáveis? Não.

Linha de execução 6:
-Nós OEO posicionados: 9, 2, 8.
- Todos os pares de nós (*o,d*) são alcançáveis? Não.

Sequência do posicionamento de nós:



Linha de execução 7:
-Nós OEO posicionados: 2, 8, 3.
- Todos os pares de nós (*o,d*) são alcançáveis? Não.

Linha de execução 8:
- Nós OEO posicionados: 9, 8, 2.
- Todos os pares de nós (*o,d*) são alcançáveis? Não.

Passos 5 e 6 – Avaliação das possíveis soluções com auxílio do algoritmo GASTL	
Solução 1 – Nós OEO posicionados: 2, 4, 6. Solução 2 – Nós OEO posicionados: 2, 4, 8.	5. Para cada solução, - calcular as rotas viáveis de menor caminho (rotas candidatas); - executar o algoritmo GASTL, a partir das rotas candidatas; - retornar rota e solução de posicionamento que obtiveram melhor avaliação pela função de avaliação do GASTL. 6. O Pico da Frequência de Uso dos Enlaces das rotas encontradas pelo algoritmo GASTL para as soluções de posicionamento de nós OEO, Solução 1 e Solução 2, foi de 28 e 22, respectivamente. Portanto, a **Solução 2** foi a escolhida.

Figura 3. Exemplo de passo-a-passo da estratégia IARRP.

A próxima seção apresenta os resultados de estudos de simulação que comparam a estratégia proposta IARRP com algoritmos de posicionamento de nós regeneradores OEO e algoritmos de roteamento, em termos de dimensionamento de nós OEO, probabilidade de bloqueio e utilização da rede.

5. Avaliação de Desempenho

A ferramenta de simulação TONetS [21] foi estendida para suportar estudos de avaliação de desempenho de redes ópticas translúcidas cientes da degradação da camada física óptica. Esta seção apresenta resultados de avaliação de desempenho considerando os algoritmos de posicionamento de nós OEO – NDF [13], HNF [4] e IARRP (proposto neste trabalho) – e os algoritmos de roteamento multirrestritivos – Off-MRR [11], MRR [11] e IARRP (proposto neste trabalho).

Este estudo de avaliação de desempenho segue as seguintes características básicas. A demanda de tráfego é composta por requisições de circuitos ópticos representados por pares de nós (origem, destino). A carga de tráfego é distribuída uniformemente entre todos os $N\times(N-1)$ pares de nós (origem, destino). A geração de requisições é um processo poissoniano de taxa média λ e o tempo de retenção dos circuitos é distribuído exponencialmente com média $1/\mu$; a intensidade de tráfego na rede em Erlangs é dada por $\rho=\lambda/\mu$. Todos os enlaces da rede são bidirecionais e possuem 40 comprimentos de onda em cada sentido. O algoritmo *First-Fit* [9] é utilizado na alocação dos comprimentos de onda. Para cada simulação são realizadas quatro replicações com diferentes sementes de geração de valores aleatórios. São geradas duas milhões de requisições para cada replicação. Os resultados gráficos apresentam os intervalos de confiança com um nível de confiança de 95%.

Neste estudo de avaliação de desempenho, a estratégia IARRP é comparada tanto com algoritmos de posicionamento de nós OEO, como algoritmos de roteamento, uma vez que a estratégia IARRP propõe o posicionamento de nós OEO e o roteamento, simultaneamente. As topologias avaliadas neste estudo foram as topologias das redes Abilene e Americana, representadas nas Figuras 4a e 4b, respectivamente. A estratégia IARRP é comparada com os algoritmos de posicionamento de nós regeneradores NDF e HNF (sob as métricas dimensionamento de nós regeneradores OEO, probabilidade de bloqueio e utilização da rede) e com os algoritmos de roteamento multirrestritivo Off-MRR e MRR (sob as métricas de probabilidade de bloqueio e utilização da rede.

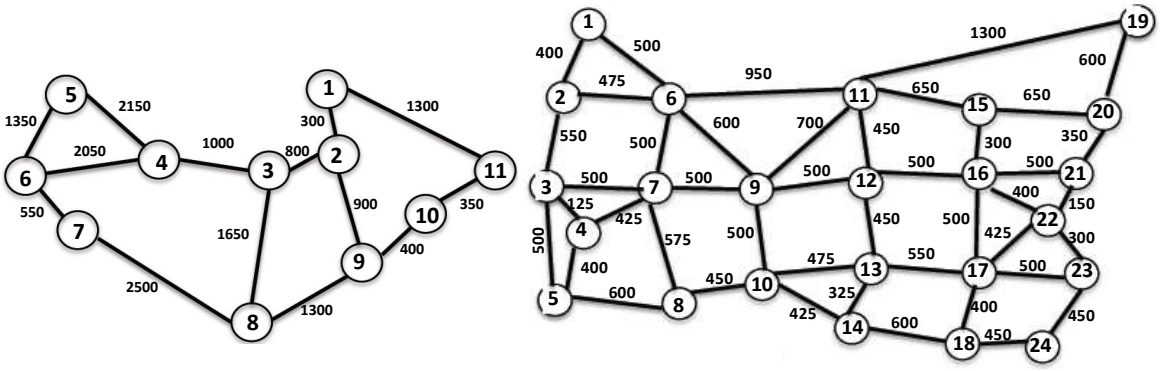


Figura 4. Topologias de rede avaliadas.

A Tabela I mostra os parâmetros para o algoritmo genético GASTL utilizado como módulo na estratégia IARRP nos experimentos deste trabalho.

Tabela I. Parâmetros utilizados para o algoritmo genético GASTL.

Parâmetro	Valor	Parâmetro	Valor
Probabilidade de cruzamento	0,9	Número de Gerações	10000
Número de pontos de cruzamento	2	Porcentagem de Elitismo	0,01
Probabilidade de Mutação	0,005	Função de Avaliação (*fitness*)	MP (para a topologia Abilene) e MDP (para a topologia Americana)
Tamanho da População	1000		

As restrições impostas foram as seguintes: distância (2.500 km para a topologia Abilene e 1.300 km para a topologia Americana), *Link Crosstalk* (2 unidades) [11] e saltos (2 saltos). O valor para a restrição de distância em cada topologia foi imposto de forma a tornar cada enlace da rede utilizável, ou seja, o limite de distância corresponde, no mínimo, à distância do maior enlace da rede.

A próxima subseção apresenta os resultados de dimensionamento de nós regeneradores OEO, comparando o posicionamento de nós OEO fornecido pelos algoritmos IARRP, NDF e HNF. Por fim, a última subseção discute os resultados de probabilidade de bloqueio e utilização da rede apresentados pelas estratégias de roteamento IARRP, Off-MRR e MRR.

5.1. Dimensionamento e Posicionamento de Nós Regeneradores OEO

A Tabela II apresenta, para cada topologia utilizada neste estudo, a quantidade de nós OEO posicionados por cada estratégia de posicionamento avaliada neste estudo.

Tabela II. Desempenho dos algoritmos de posicionamento de nós OEO.

Topologia de Rede	Estratégia de Posicionamento	Quantidade de Nós OEO Necessários	Nós OEO Posicionados
Abilene	NDF	3	2,4,8.
	HNF	3	2,4,6.
	IARRP	3	2,4,8.
Americana	NDF	8	6, 16, 3, 9, 13, 8, 11, 17.
	HNF	5	9, 16, 7, 11, 17.
	IARRP	4	11, 13, 7, 17.

Na Tabela II é possível observar que, para a topologia Abilene, os algoritmos de

posicionamento apresentaram o mesmo número de nós regeneradores OEO. No entanto, para a topologia Americana, o algoritmo NDF apresentou o pior desempenho em termos de quantidade de nós regeneradores OEO necessários. O algoritmo NDF escolhe os nós que apresentam maior grau físico, sem nenhum outro critério de escolha. Alguns desses nós com maior número de interfaces podem estar muito próximos topologicamente, necessitando de mais iterações do algoritmo para abranger toda a rede e, consequentemente, tornar todo par de nós (origem, destino) alcançável.

Por outro lado, é possível observar (Tabela II) que no cenário da topologia de rede Americana, a estratégia IARRP posicionou 4 (quatro) nós OEO, enquanto que as estratégias HNF e NDF posicionaram 5 (cinco) e 8 (oito) nós OEO, respectivamente. A estratégia proposta IARRP apresentou desempenho igual ou superior às demais estratégias avaliadas, em todos os cenários estudados (incluindo outras topologias de rede que, por limitação de espaço, não foram apresentados neste artigo). A superioridade do posicionamento realizado pela estratégia IARRP é justificada pelo fato dela utilizar a estratégia HNF como módulo de iteração, permitindo durante essa iteração a escolha de nós OEO diferentes da estratégia HNF, através da técnica de *backtracking*.

5.2. Probabilidade de Bloqueio e Utilização

Para a análise da probabilidade de bloqueio e utilização da rede, o algoritmo IARRP foi comparado com os outros dois algoritmos de roteamento multirrestritos Off-MRR e MRR. Vale resaltar que a estratégia híbrida IARRP proposta tem como objetivo o posicionamento de nós regeneradores OEO e roteamento ao mesmo tempo. Desta forma, foi utilizado o algoritmo HNF para o posicionamento de nós OEO nos cenários das estratégias de roteamento MRR e Off-MRR, denotados nos gráficos por HNF-MRR e HNF-OffMRR, respectivamente.

As Figuras 5 e 6 mostram os gráficos de Probabilidade de Bloqueio e Utilização da Rede para as topologias ilustradas na Figura 4a e 4b, respectivamente. Como evidenciam as Figuras 5a e 6a, o algoritmo IARRP apresentou os melhores resultados de probabilidade de bloqueio, em todos os cenários avaliados.

Conforme estudo realizado em [11] e comprovados pelos resultados apresentados neste trabalho, é esperado que o algoritmo Off-MRR não apresente menor probabilidade de bloqueio quando comparado ao algoritmo MRR, pois o algoritmo Off-MRR não considera em seu cálculo as restrições online como, por exemplo, a restrição imposta de *Link Crosstalk*. Além disso, o algoritmo Off-MRR retorna a primeira rota viável encontrada na fase de planejamento da rede.

Por outro lado, apesar de o algoritmo proposto IARRP considerar apenas as restrições estáticas (*offline*), ele realiza balanceamento de carga através do balanceamento da Frequência de Uso dos enlaces da rede, pois, dentre todas as menores rotas viáveis, escolhe de maneira criteriosa a rota fixa para atendimento das requisições dinâmicas de circuito óptico, fazendo uso do algoritmo genético GASTL apresentado na Seção 3. Já o algoritmo MRR, apesar de buscar atender tanto às restrições *offline* como às restrições online, retorna apenas a primeira rota viável encontrada dinamicamente.

Analisando os gráficos de utilização da rede (Figuras 5b a 6b), é possível concluir que o algoritmo IARRP apresenta melhor distribuição da utilização da rede, pois os gráficos mostram uso equivalente de recursos da rede por parte de todos os

algoritmos e menor probabilidade de bloqueio por parte do algoritmo IARRP.

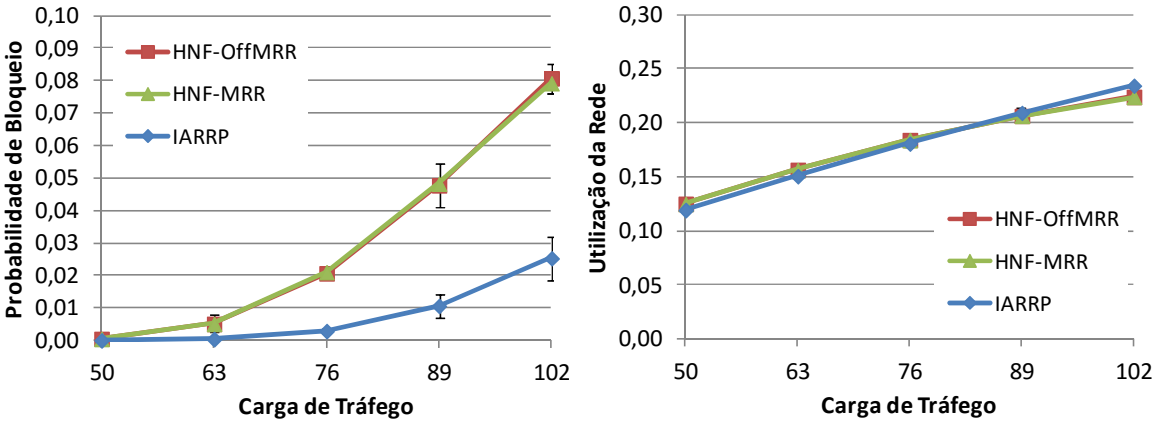


Figura 5. PB e Utilização – Topologia de Rede Abilene.

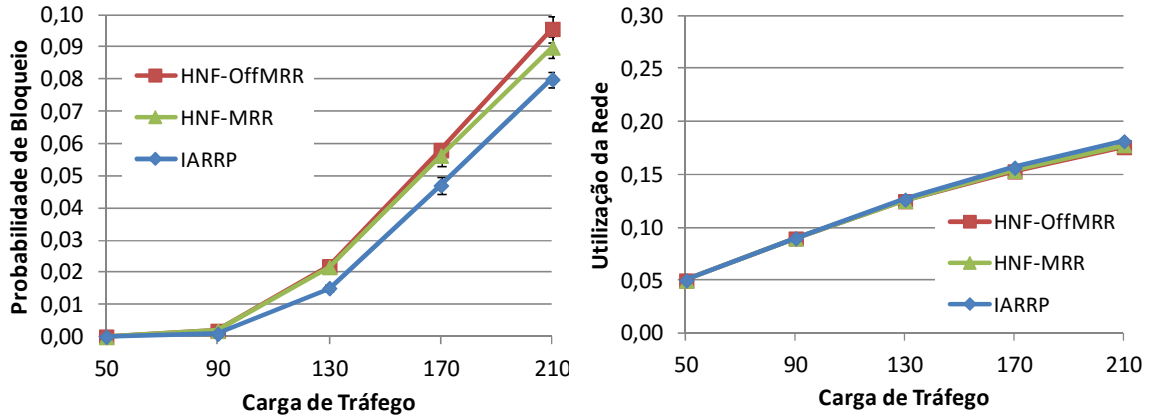


Figura 6. PB e Utilização – Topologia da Rede Americana.

A Tabela III mostra os valores de mínimo, máximo, média e desvio padrão da utilização dos enlaces das redes estudadas por parte dos algoritmos Off-MRR, MRR e IARRP para a carga máxima de tráfego exibida nos gráficos das Figuras 5 e 6. Ao lado do valor da utilização mínima e máxima está indicado o enlace que apresentou tal utilização, denotado por <*a,b*> (enlace do nó *a* para o nó *b*).

Tabela III. Valor mínimo, máximo, média e desvio padrão da utilização dos enlaces por parte dos algoritmos de roteamento multirrestritivos.

Topologia de Rede	Algoritmo	Média	Desvio Padrão	Mínimo	Máximo
Abilene	Off-MRR	0,2343	0,1685	0,0231 *Enlace <3,8>*	0,5672 *Enlace <2,3>*
	MRR	0,2237	0,1683	0,0228 *Enlace <7,8>*	0,5660 *Enlace <2,3>*
	IARRP	0,2343	0,1521	0,0459 *Enlace <6,7>*	0,4799 *Enlace <2,3>*
Americana	Off-MRR	0,1759	0,1597	0,0094 *Enlace <5,4>*	0,6245 *Enlace <13,17>*
	MRR	0,1777	0,1605	0,0094 *Enlace <5,4>*	0,6271 *Enlace <13,17>*
	IARRP	0,1813	0,1434	0,0187 *Enlace <3,4>*	0,5708 *Enlace <11,12>*

Na Tabela III, nota-se maior média de utilização dos enlaces, no entanto, menor desvio padrão, por parte do algoritmo IARRP (roteamento *offline*), inclusive quando comparado ao algoritmo alternativo-adaptativo MRR (roteamento adaptativo), ratificando a eficiência do balanceamento de carga realizado na fase de planejamento da rede por parte do algoritmo IARRP.

Vale ressaltar ainda que durante a fase de operação da rede a complexidade computacional do algoritmo IARRP é a mesma do algoritmo Off-MRR, pois ambos escolhem suas rotas na fase de planejamento da rede, sendo mínimo o tempo de execução do algoritmo na fase dinâmica da rede. Nestas simulações, o tempo médio de execução dos algoritmos IARRP e Off-MRR foi menor que 0,001 milissegundo, enquanto que o tempo médio de execução do algoritmo MRR foi de pouco mais de 2 milissegundos em todos os cenários avaliados.

6. Considerações Finais

Este trabalho propôs uma nova estratégia (IARRP) para solucionar, simultaneamente, tanto o problema de posicionamento de nós regeneradores OEO como também o problema de roteamento multirrestritivo em redes ópticas translúcidas. A nova estratégia proposta, IARRP, foi avaliada comparativamente com outros algoritmos propostos na literatura, em diversas topologias de rede. Por limitação de espaço, foram apresentados resultados para duas topologias (Abilene e Americana). Nestas topologias de rede, bem como nas outras topologias de rede avaliadas, o IARRP apresentou os melhores resultados sob as métricas de dimensionamento de nós OEO, probabilidade de bloqueio e utilização dos recursos da rede.

Além disso, destaca-se que, enquanto a estratégia MRR é uma estratégia alternativa adaptativa (dinâmica), a estratégia proposta IARRP é uma estratégia de roteamento *offline* (fixo), ou seja, apresenta o mínimo de complexidade possível durante a operação da rede e, ainda assim, supera as expectativas, apresentando melhor desempenho, em todos cenários avaliados. Tal comportamento é justificado pela eficiência no balanceamento de carga por parte da estratégia proposta IARRP.

Referências

[1] T. Morioka, “New generation optical infrastructure technologies: ‘EXAT initiative’ towards 2020 and beyond” *in Proc. 14th OECC*, Jul. 2009, p. 1–2.

[2] F. Ye, *et al*, “Wavelength-Dependence of Inter-Core Crosstalk in Homogeneous Multi-Core Fibers”, *in Photonics Technology Letters, IEEE* , v. 28, n.1, p.27-30, Janeiro, 2016.

[3] M. Gagnaire e S. Zahr. “Impairment-Aware Routing and Wavelength Assignment in Translucent Optical Networks: State of the Art”. *IEEE Communication Magazine*, v. 47, n. 5, p. 55-61, Maio, 2009.

[4] I. Nath, M. Chatterjee e U. Bhattacharya, “A survey on regenerator Placement Problem in translucent optical network”, *in International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, p.408-413, 4-5, April, 2014.

[5] G. Shen e R. S. Tucker. “Translucent Optical Networks: The Way Forward”, *IEEE Communications Magazine*, v. 45, n. 2, p. 48-54, Fevereiro, 2007.

[6] G. M. Durães, *et al.*, “The choice of the best among the shortest routes in transparent

optical networks", *Computer Networks*, v. 54, p. 2400-2409, 2010.

[7] B. Ramamurthy, *et al*, "Impact of Transmission Impairments on the Teletraffic Performance of Wavelength-routed Optical Networks", *Journal of Lightwave Technology (JLT)*, v. 17, n. 10, p. 1713- 1723, 1999.

[8] S. Azodolmolky, *et al*. "A survey on physical layer impairments aware routing and wavelength assignment algorithms in optical networks", *Computer Networks*, p. 926-944, Janeiro, 2009.

[9] S. Varanasi, S. Bandyopadhyay e A. Jaekel, "Impairment aware dynamic routing and wavelength assignment in translucent optical WDM networks," in *Distributed Computing and Networking, ser. Lecture Notes in Computer Science*, Springer Berlin Heidelberg, v. 8314, p. 363–377, 2014.

[10] A. G. Rahbar, "Review of dynamic impairment-aware routing and wavelength assignment techniques in all-optical wavelength-routed networks", *IEEE Communications Surveys & Tutorials*, v.14, n.4, p. 1065-1089, 2012.

[11] G. M. Durães, J. A. S. Monteiro e W. F. Giozza, "A new approach for multi-restrictive routing in translucent optical networks" in *2014 Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, Florianópolis, Brazil, p. 428-435, 5-9, Maio 2014.

[12] G. Shen, *et al*., "Sparse placement of electronic switching nodes for low blocking in translucent optical networks". *Journal of Optical Networking*, v. 1, n. 12, p. 424-441, Dezembro, 2002.

[13] G. Shen e W. D. Grover, "Segment-based approaches to survivable translucent network design under various ultra-long-haul system reach capabilities". *Journal of Optical Networking*, v. 3, n. 1, p. 1-24, Janeiro, 2004.

[14] A. Sen e S. Murthy e S. Bandyopadhyay, "On Sparse Placement of Regenerator Nodes in Translucent Optical Networks", IEEE GLOBECOM 2008, p. 1-6, Dezembro, 2008.

[15] F. Martinelli, *et al.*, "Genetic approach for optimizing the placement of all-optical regenerators in WSON", *in IEEE/OSA Journal of Optical Communications and Networking*, v. 6, n. 11, p.1028-1037, Novembro, 2014.

[16] E. Gourdin, *et al*., *Facility Location: Applications and Theory*, Springer, 2001.

[17] B. Garcia-Manrubia, "Offline Impairment-Aware RWA and Regenerator Placement in Translucent Optical Networks", *in Journal of Lightwave Technology*, v.29, n.3, p.265-277, Fevereiro, 2011.

[18] X. Yang e B. Ramamurthy, "Sparse Regeneration in Translucent Wavelength-Routed Optical Networks: Architecture, Network Design and Wavelength Routing", *Photonic Network Communications*, v.1, n, 10, p. 39–53, Março, 2005.

[19] D. A. R. Chaves, *et al.*, "Novel strategies for sparse regenerator placement in translucent optical networks". *Photonic Network Communications*, v.1, n. 24, p. 237–251, Maio, 2012.

[20] J. H. Holland, "Adaptation in Natural and Artificial Systems", *University of Michigan Press, Ann Arbor, Michigan*, 1975/1992.

[21] A. C. B. Soares, G. M. Durães, W. Giozza e P. Cunha, "TONetS: Ferramenta para Avaliação de Desempenho de Redes Ópticas Transparentes" *in VII Salão de Ferramentas do Simpósio Brasileiro de Redes de Computadores - SBRC*, Maio 2008.

Nova Solução para o Problema de Roteamento em Redes Ópticas Elásticas utilizando Algoritmo Ciente de Fragmentação Baseado em Sistemas Fuzzy

Ítalo Barbosa Brasileiro[1], **José Valdemir dos Reis Júnior**[1], **André Castelo Branco Soares**[1]

[1]Departamento de Computação – Universidade Federal do Piauí (UFPI)
Teresina – PI – Brazil

italo.barbosabrasileiro@yahoo.com.br,{andre.soares,valdemirreis}@ufpi.edu.br

Abstract. *This paper presents a new solution to the routing problem in Elastic Optical Networks using an intelligent approach based on Fuzzy System. The proposed algorithm defines the best route to a given pair, considering as metrics the fragmentation of routes, the number of occupied slots and the index of the first free interval to establish the circuit. Simulations taken with this new approach consider the USA and NSFNet topologies, which has large area and a crescent number of users. The obtained results shows that the proposed approach has blocking 30,43% lower when compared to DJK (Dijkstra), 20,87% lower to another Fuzzy implementation and 13,24% lower to MMRDS in USA topology, and 95,45% lower when compared to DJK, 78,03% compared to another Fuzzy implementation and 63,81% lower to MMRDS in NSFNet topology. The performance stated by this approach enables the exploration of new intelligent techniques to improve the quality of service in Elastic Optical Networks.*

Resumo. *Neste artigo é apresentada uma nova solução para o problema de roteamento em Redes Ópticas Elásticas utilizando uma abordagem inteligente baseada em Sistema Fuzzy. O algoritmo proposto define a melhor rota para um dado par, considerando como métricas a fragmentação das rotas, a quantidade de slots livres e o índice do primeiro intervalo livre para o estabelecimento do circuito. As simulações para essa nova abordagem consideram as topologias USA e NSFNet, que ocupam grande área e possuem número crescente de usuários. Os resultados obtidos apontam que a abordagem proposta apresenta bloqueio 30,43% inferior ao DJK (Dijkstra), 20,87% inferior a outra implementação Fuzzy e 13,24% inferior ao MMRDS para a topologia USA, enquanto que para a topologia NSFNet apresenta bloqueio 95,45% inferior ao DJK, 78,03% inferior à outra implementação Fuzzy e 63,81% inferior ao MMRDS. Os desempenhos relatados para esta proposta possibilitam a exploração de novas técnicas inteligentes para melhorar a qualidade de serviço das Redes Ópticas Elásticas.*

1. Redes Ópticas Elásticas

Para acompanhar a crescente necessidade de maiores taxas de transmissão, surgem tecnologias que permitem a transmissão de grande volume de dados. Dentre essas tecnologias estão as Redes Ópticas Elásticas (EON - *Elastic Optical Networks*) [Kretsis et al. 2014],

que utilizam a luz como meio de transmissão dos dados, conseguindo atingir altas taxas de transmissão. As Redes Elásticas permitem o estabelecimento de múltiplos circuitos em uma única fibra, por meio da divisão do espectro em intervalos de frequência, aumentando a capacidade de transmissão [Moura et al. 2014].

As EON possuem a vantagem de acomodar circuitos que exigem diferentes quantidades de largura de banda, de forma a não subutilizar o intervalo espectral selecionado. Isso se deve à organização espectral: o intervalo de frequência do espectro é dividido em pequenos intervalos de frequência média de 12,5 GHz denominados *slots* [Cugini et al. 2013]. A Figura 1 ilustra os slots do espectro óptico de uma fibra. Para atender requisições que necessitem de maiores larguras de banda, slots adjacentes são agrupados para formar um intervalo espectral maior. Esse intervalo é destinado aos circuitos que necessitam de maior quantidade de banda para serem estabelecidos. Na Fig. 1 é possível observar a existência de circuitos com 1, 2 ou 3 slots. Essa ”elasticidade”do espectro óptico é o que caracteriza e concede o nome das Redes Ópticas Elásticas.

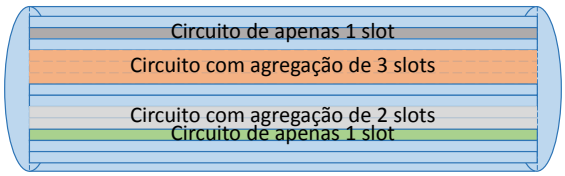


Figura 1. Representação de fibra óptica e capacidade elástica de slots.

Para o uso eficiente de redes ópticas, é necessária a seleção de rota e faixa espectral para atender as requisições de circuito, de modo a reduzir o impacto causado pelos recursos alocados na taxa de QoS (*Quality of Service*) causado pelos recursos a serem alocados. Portanto, o problema RSA (*Routing and Spectrum Allocation*) [Gerstel et al. 2012] torna-se um desafio importante para a utilização de Redes Elásticas. A rota e a faixa espectral ideal escolhida como solução para o problema RSA devem resultar em melhor utilização dos recursos da rede e preservar a organização dos circuitos já estabelecidos.

Para manter a transmissão dos dados ao longo dos enlaces é necessário cumprir alguns requisitos provenientes do meio óptico, denominados de *restrições de continuidade e contiguidade espectral* [Wang and Mukherjee 2014]. Na *restrição de continuidade*, para que não haja necessidade de conversão do sinal óptico para o domínio eletrônico, torna-se obrigatória a permanência do sinal óptico no mesmo intervalo espectral entre os nós de origem e destino. Assim, ao alocar um conjunto de slots para a formação de um circuito, o mesmo conjunto deve estar disponível em todos os enlaces constituintes da rota selecionada. Na *restrição de contiguidade*, para atender uma requisição, é preciso alocar um conjunto de slots dispostos de forma adjacente no espectro óptico. Assim, utiliza-se apenas um transmissor para cada circuito, pois apenas uma faixa espectral será ocupada. A Figura 2 ilustra os problemas de *continuidade* e *contiguidade* espectral.

Devido a restrições de *continuidade* e *contiguidade*, e a medida que circuitos são estabelecidos e desligados, vão surgindo pequenas lacunas de slots livres no espectro. Na Fig. 2(a), observa-se que algumas lacunas não poderão ser preenchidas devido à restrição de continuidade. Na Fig. 2(b), a restrição de contiguidade impede o estabelecimento de circuitos maiores que 2 slots, pois todo o espectro está dividido em pequenos fragmentos. A presença desses intervalos, em maior proporção, interfere no funcionamento da

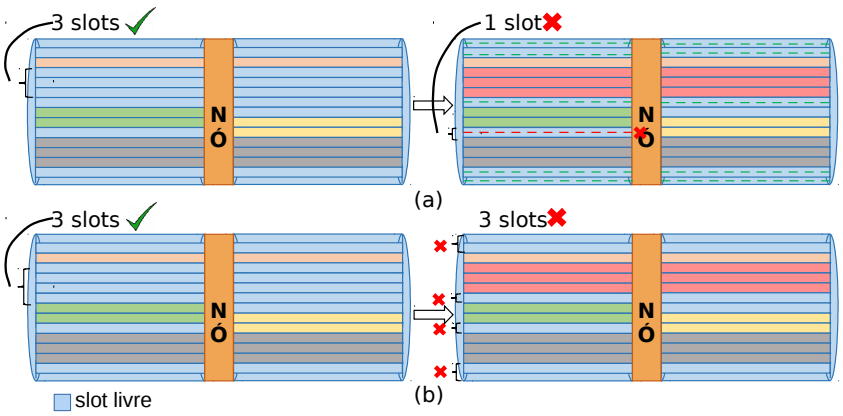


Figura 2. Restrições de *continuidade* (a) e *contiguidade* (b).

rede, pois algumas requisições não serão atendidas, ainda que existam slots suficientes. Estes slots estarão dispersos no espectro óptico, impedidos de serem alocados devido à *restrição de contiguidade*. Esse problema surge juntamente com as Redes Elásticas, e é caracterizado como *Problema de Fragmentação* [Wang and Mukherjee 2014].

Considerando o problema RSA, juntamente com suas restrições e o agravante de fragmentação, no presente trabalho é proposto um novo algoritmo baseado em lógica *Fuzzy* para solucionar o problema de roteamento em Redes Ópticas Elásticas. A técnica proposta utiliza informações do nível de fragmentação do espectro óptico, a fim de escolher a rota com menor grau de fragmentação para a formação de novos circuitos. Além de ser um sistema ciente de fragmentação, o algoritmo também utiliza informações como a quantidade de slots ocupados e o índice do slot que encabeça o primeiro intervalo capaz de suportar a requisição, para viabilizar um melhor balanceamento de carga durante a escolha de rotas.

A organização do trabalho segue a seguinte orientação: a Seção 2 apresenta uma visão geral dos trabalhos relacionados; a Seção 3 apresenta a modelagem do sistema *Fuzzy* proposto; a Seção 4 apresenta uma avaliação de desempenho do algoritmo, comparado com outros algoritmos de roteamento da literatura; e a Seção 5 apresenta as conclusões.

2. Trabalhos Relacionados

Sistemas de comunicação que utilizam fibra óptica como meio de transmissão oferecem grande escalabilidade para suportar o crescente tráfego de dados na Internet, devido à grande disponibilidade de largura de banda [Gerstel et al. 2012]. Essa característica torna-se evidente a medida em que se aprofunda a pesquisa em Redes Ópticas, iniciada comercialmente em 1980, quando se registravam transmissões próximas a 45Mb/s [Alferness 2012] [Tomkos et al. 2012], até atualmente, quando é constatado transmissões de 20Tb/s em uma fibra óptica [Cai et al. 2012].

Para melhor gerenciamento dos recursos da rede é necessário definir um conjunto de rotas e intervalos de espectro a serem alocados para as diferentes requisições de estabelecimento de circuito óptico. Essa necessidade constitui o problema RSA (*Routing and Spectrum Allocation*), que vem se tornando uma importante área de pesquisa na literatura relacionada [Yin et al. 2013].

Os algoritmos que atacam o problema RSA podem ser classificados em dois tipos: *integrados* e *sequenciais*. Os algoritmos *sequenciais* dividem em duas partes o problema RSA, e podem ser classificados em algoritmos de roteamento [Wan et al. 2012] ou algoritmos de alocação de espectro [Wright et al. 2015]. Já os *integrados* tentam resolver o problema RSA em uma única etapa, realizando simultaneamente a escolha da rota e do intervalo de slots destinado para a requisição [Wang and Mukherjee 2012]. O algoritmo proposto neste trabalho ataca o problema de roteamento, portanto é classificado como *sequencial*.

Além do problema RSA, é observado o problema de *Fragmentação* nas redes elásticas [Shakya and Cao 2013]. A medida em que circuitos de diferentes larguras de banda são estabelecidos e desligados, pequenas lacunas de slots não-alocados surgem entre intervalos de slots ocupados. Esse problema pode evoluir de forma a impedir o atendimento de requisições, mesmo que a rota contenha a quantidade suficiente de slots. A fim de evitar a ocorrência desse problema, é feito o estudo de uma classe de algoritmos, classificados como *cientes de fragmentação*. Os autores em [Cugini et al. 2013] apresentam uma técnica de desfragmentação espectral, na qual ocorre suspensão dos circuitos, durante alguns segundos, para reorganização dos mesmos no espectro óptico. Já em [Wang and Mukherjee 2014] é apresentada uma política de particionamento do espectro, a fim de reduzir a desigualdade de atendimento para requisições de diferentes larguras de banda, impactando diretamente na taxa de fragmentação da rede. Nesse sentido, a solução RSA aqui proposta também busca reduzir o impacto da fragmentação no processo de estabelecimento de circuitos, pois leva em consideração o grau de fragmentação das rotas candidatas para a escolha da rota adequada.

O estudo do problema de roteamento é essencial para garantir uma boa utilização dos recursos de redes. Há algum tempo que esse problema é avaliado [Dijkstra 1959], e sempre que surgem novas tecnologias para redes, é preciso reformulá-lo considerando o novo conjunto de restrições e variáveis. Como uma solução para o problema de roteamento em Redes Ópticas, em [Santos et al. 2012] é proposto o algoritmo MMRDS (*Melhor entre as Menores Rotas com Decisão por Similaridade*), que considera a similaridade entre as rotas de menor caminho para cada par de nós, alocando aquelas que permitam um melhor balanceamento entre todos os enlaces da rede. Os algoritmos de Dijkstra [Dijkstra 1959] e MMRDS são simulados e comparados com a abordagem, utilizando Sistema *Fuzzy*, aqui proposta.

Algumas propostas para solucionar problemas em Redes Ópticas são baseadas em lógica *Fuzzy*. Em [Kavian et al. 2010] foi proposto um modelo *Fuzzy* em programação linear para planejamento de capacidade em redes ópticas. Em [Bhanja and Mishra 2015], foi modelado um sistema *Fuzzy* para atacar o problema RSA Dinâmico em Redes Ópticas WDM, e a seleção do caminho de menor custo foi baseada em alguns critérios provenientes da camada física, como latência e distância real dos enlaces. Já em [Ribeiro 2014] foi proposta uma técnica de roteamento baseada em lógica *Fuzzy* que verifica a quantidade de slots ocupados e o tamanho dos enlaces para a seleção de rotas. Esta técnica será avaliada neste trabalho, juntamente com DJK e MMRDS. Em [dos Reis et al. 2015] foi proposto um sistema *Fuzzy*, modelado para mitigar a variação de temperatura nos enlaces de transmissão em redes OCDMA (Optical Code-Division Multiple-Access).

A proposta deste artigo ataca o problema de roteamento utilizando um sistema

Fuzzy na escolha da rota adequada entre as rotas candidatas para o estabelecimento do circuito. Na seção seguinte será demonstrada a modelagem do algoritmo proposto.

3. Roteamento por Lógica Fuzzy

Para a escolha da rota durante a fase operacional da rede, o algoritmo proposto utiliza um sistema *Fuzzy* ao avaliar as rotas candidatas. A Figura 3 apresenta um fluxograma da execução do algoritmo. A chegada de requisições ocorre de forma dinâmica entre todos os pares de nós de origem e destino da rede, e as rotas candidatas são aquelas de menor caminho, caso exista mais de uma rota de menor caminho.

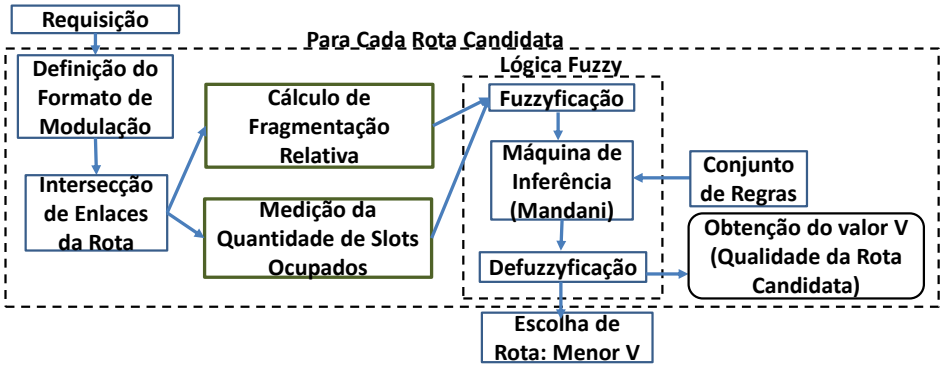


Figura 3. Fluxograma do algoritmo proposto utilizando Sistemas *Fuzzy*.

Durante a fase *offline* do algoritmo (que ocorre antes da fase operacional da rede), são calculados os menores caminhos para todos os pares de nós da rede. Na fase *online*, ao surgir uma requisição, é verificado qual os nós de origem e de destino envolvidos. Após, busca-se o conjunto de menores caminhos para o par, e cada um dos menores caminhos passa por um processo *Fuzzy* a fim de se encontrar um valor V de qualidade da rota.

O sistema *Fuzzy* permite inferir uma *taxa de pertinência* do elemento em um conjunto, através da utilização de variáveis linguísticas e regras de controle. As variáveis classificam o elemento, enquanto as regras reunem as diferentes classificações e definem uma qualidade que represente o elemento avaliado. Considerando essa característica, percebe-se a diferença de procedimentos *Fuzzy* para a lógica clássica, na qual um elemento não-pertence (0) ou pertence totalmente (1) a um conjunto.

Para o cálculo de pertinência de uma rota, é feito inicialmente o processo de *Fuzzyficação*, quando é verificada um conjunto de características da rota em questão. Cada característica é avaliada em uma *função de pertinência*, que permite inferir quais variáveis linguísticas qualificam a rota em estudo. Após, o conjunto de variáveis linguísticas encontrado é transmitido para o *modelo de inferência de Mamdani* [Iancu 2012], que juntamente com o conjunto de regras de base define qual variável linguística de cada critério avaliado representa de melhor forma a rota. Por fim, ocorre o processo de *Defuzzyficação*, em que se obtém um valor numérico V para caracterizar o estado da rota em estudo.

O primeiro critério utilizado como função de pertinência é a *Quantidade de Slots Ocupados* na rota. Slots ocupados são aqueles que não estão livres de forma contínua na rota, ou seja, os slots que são utilizados em pelo menos um enlace da rota. Rotas com

muitos slots ocupados devem ser evitadas, a fim de não criar os chamados *gargalos* na rede. A Figura 4 apresenta uma modelagem da função de pertinência para a avaliação da *Quantidade de Slots Ocupados*. As variáveis linguísticas definidas foram *q1*, *q2*, *q3*, *q4* e *q5*, que representam respetivamente rotas com distribuição Gaussiana de média igual a 0, 100, 200, 300 e 400 slots ocupados e desvio igual a 80.

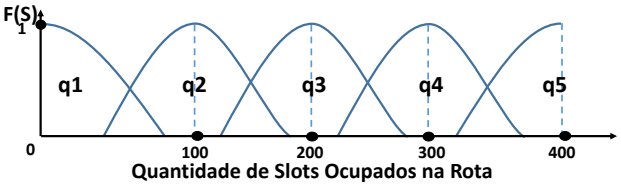


Figura 4. Função de Pertinência para Quantidade de Slots Ocupados.

O segundo critério utilizado como função de pertinência é o grau de *Fragmentação Relativa* [Horota et al. 2014] da rota. A *Fragmentação Relativa* permite verificar o quanto uma rota se encontra fragmentada para atender requisições que necessitam de determinada quantidade de slots. A Equação 1 apresenta o cálculo da *Fragmentação Relativa*:

$$F(c) = \frac{c * RS(c)}{S_L}, \tag{1}$$

na qual c é a quantidade de slots da requisição a ser atendida, $RS(c)$ é a quantidade de requisições simultâneas de tamanho c que podem ser atendidas considerando o estado corrente do espectro óptico e S_L é a quantidade total de slots livres de forma contínua nos enlaces da rota observada. A Figura 5 exemplifica o cálculo de *Fragmentação Relativa* considerando uma rota de dois enlaces.

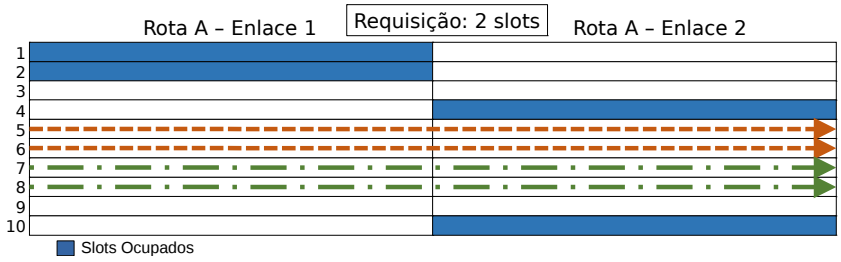


Figura 5. Exemplificação da métrica de Fragmentação Relativa.

No exemplo ilustrado na Figura 5, considerando uma requisição que necessite de 2 slots para ser atendida, percebe-se que apenas dois conjuntos de 2 slots contínuos e contíguos podem ser alocados de forma simultânea considerando o estado corrente do espectro. Observa-se também que, além dos dois conjuntos de slots que podem ser alocados, restam mais 2 slots que estão livres, mas não podem ser alocados devido à fragmentação da rota em questão. Portanto, o grau de *Fragmentação Relativa* da rota ilustrada para requisições de 2 slots é (2 * 2) / 6 = 0.6667.

Avaliar a *Fragmentação Relativa* é útil para apontar as rotas de menor grau de fragmentação. Utilizar essa métrica como critério de avaliação permite verificar as rotas

mais propícias para atender requisições que exijam diferentes quantidades de largura de banda. A Figura 6 ilustra a função de pertinência para a *Fragmentação Relativa*, dividida em 5 conjuntos *Fuzzy* com formato gaussiano: baixo, médio baixo, médio, médio alto e alto, de valor médio respectivamente igual a 0, 0.25, 0.5, 0.75 e 1, e desvio igual a 0.2.

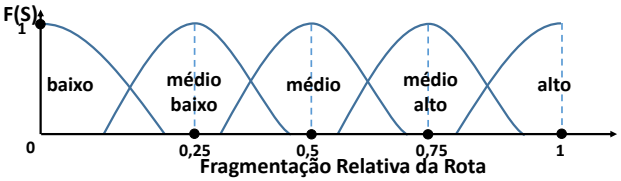


Figura 6. Função de pertinência para o critério de Fragmentação Relativa.

O terceiro critério utilizado como função de pertinência é o *Índice do Slot Inicial do Primeiro Intervalo Espectral Alocável*. Esse critério favorece a escolha da rota mais próxima do início espectral, acarretando em melhor organização no espectro utilizado. A escolha de rotas com o primeiro intervalo de slots alocáveis mais próximo do início do espectro viabiliza melhor aplicação da política *First Fit* no processo de alocação espectral. A Figura 7 apresenta a modelagem da função de pertinência para o critério de *Índice do Slot Inicial do Primeiro Intervalo Espectral Alocável*, na qual $q1$ apresenta a variável de maior qualidade e $q5$ a de menor qualidade.

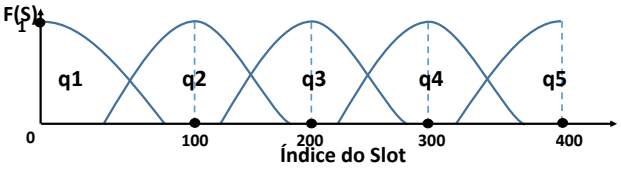


Figura 7. Função de Pertinência para o critério de Índice do Slot Inicial do Primeiro Intervalo Espectral Alocável.

Após a parametrização das variáveis linguísticas para o processo de *Fuzzyficação*, é necessário modelar uma função que qualifique a rota após a avaliação das variáveis linguísticas que a caracterizam. A Figura 8 apresenta a modelagem das variáveis linguísticas para o processo de *Defuzzyficação*. Após a *Defuzzyficação* é obtido o valor V que será utilizado para a seleção da rota. Na modelagem deste trabalho, a rota de menor valor V dentre as rotas candidatas será a rota escolhida para o atendimento da requisição.

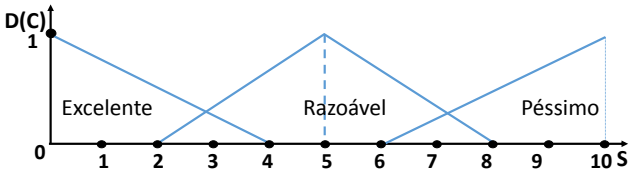


Figura 8. Conjunto de variáveis linguísticas para processo de *Defuzzyficação*.

Foram utilizadas 5 variáveis linguísticas para caracterizar cada uma das funções de pertinência. Maiores quantidades de variáveis linguísticas associariam maior comple-

xidade ao problema, considerando que um maior intervalo de valores precisaria ser observado. Já a utilização de um conjunto menor de variáveis linguísticas causaria redução impactante no grau de percepção das funções de pertinência.

Com a finalidade de agregar todas as variáveis linguísticas em um único conjunto, é utilizado o método de Mamdani no processo de inferência. A Figura 9 ilustra o funcionamento do método de Mamdani, observando os critérios de *Fragmentação Relativa* e *Quantidade de Slots Ocupados*, com valores de 0.6 e 150 respectivamente, em um total de 400 slots. Inicialmente, é definido o valor de pertinência (eixo y das Figs. 4, 6 e 7) para cada função de pertinência da rota. Nos casos em que uma rota é classificada por mais de uma variável linguística, a variável de menor valor é assumido. Após a decisão do valor para cada função de pertinência, o valor V final escolhido para a rota é o maior dentre os valores resultantes das funções de pertinência na fase anterior.

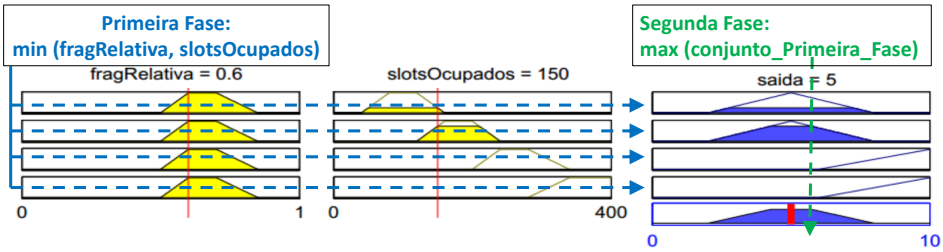


Figura 9. Conjunto de regras *Fuzzy*, com entrada de valores de Fragmentação relativa e Quantidade de Slots Ocupados iguais a 0.6 e 150, respectivamente.

Para o funcionamento da lógica *Fuzzy*, se faz necessário associar uma variável linguística do conjunto de saída a todas as possíveis combinações de variáveis linguísticas das funções de pertinência de entrada, criando assim a *base de conhecimento*. A Tabela 1 apresenta a base de conhecimento utilizada pelo algoritmo proposto.

A última etapa do algoritmo proposto é caracterizada como *Defuzzyficação*. Existem diferentes técnicas de *Defuzzyficação*, dentre as quais se considera principalmente o critério de complexidade computacional como métrica de escolha. O método utilizado neste trabalho é o Centroide (ou Centro de Massa) [Pappis and Siettos 2005]. A Equação 3 apresenta o cálculo do Centroide para se encontrar o valor V:

$$V = \frac{\sum_{s=0}^{10} S * D(S)}{\sum_{s=0}^{10} D(S)}, \tag{2}$$

na qual S representa a distribuição dos valores de classificação da rota após a *Defuzzyficação* (eixo x da Fig. 8), e $D(S)$ apresenta o valor obtido para essa classificação, após o processo de inferência. Supondo que, após a aplicação do processo de Inferência, o resultado é 0.51 para a variável linguística $excelente$, 0.23 para a variável linguística $razoavel$ e 0,0 para a variável $pessimo$, então o cálculo realizado é:

$$V = \frac{(1+2+3)*0.51 + (3+4+5+6+7)*0.23 + (7+8+9+10)*0.0}{3*0.51 + 5*0.23 + 4*0.0} = 3.287 \tag{3}$$

Tabela 1. CONJUNTO DE REGRAS DE CONTROLE DO SISTEMA FUZZY.

Qt. Slots Ocupados Frag. Rel. + Index Slot	q1	q2	q3	q4	q5
baixo & q1	excelente	excelente	excelente	excelente	razoável
baixo & q2	excelente	excelente	excelente	razoável	razoável
baixo & q3	excelente	excelente	razoável	razoável	razoável
baixo & q4	excelente	razoável	razoável	razoável	razoável
baixo & q5	razoável	razoável	razoável	razoável	péssimo
médio baixo & q1	excelente	excelente	excelente	razoável	razoável
médio baixo & q2	excelente	excelente	razoável	razoável	razoável
médio baixo & q3	excelente	razoável	razoável	razoável	razoável
médio baixo & q4	razoável	razoável	razoável	razoável	péssimo
médio baixo & q5	razoável	razoável	razoável	péssimo	péssimo
médio & q1	excelente	excelente	razoável	razoável	razoável
médio & q2	excelente	razoável	razoável	razoável	razoável
médio & q3	razoável	razoável	razoável	razoável	péssimo
médio & q4	razoável	razoável	razoável	péssimo	péssimo
médio & q5	razoável	razoável	péssimo	péssimo	péssimo
médio alto & q1	excelente	razoável	razoável	razoável	razoável
médio alto & q2	razoável	razoável	razoável	razoável	péssimo
médio alto & q3	razoável	razoável	razoável	péssimo	péssimo
médio alto & q4	razoável	razoável	péssimo	péssimo	péssimo
médio alto & q5	razoável	péssimo	péssimo	péssimo	péssimo
alto & q1	razoável	razoável	razoável	razoável	péssimo
alto & q2	razoável	razoável	razoável	péssimo	péssimo
alto & q3	razoável	razoável	péssimo	péssimo	péssimo
alto & q4	razoável	péssimo	péssimo	péssimo	péssimo
alto & q5	péssimo	péssimo	péssimo	péssimo	péssimo

Para os três critérios avaliados, utilizou-se funções de pertinência da forma gaussiana. Experimentos foram realizados com diferentes formatos de função de pertinência, mas para o algoritmo proposto a forma Gaussiana apresentou melhor resultado. Já para o processo de *Defuzzyficação*, utiliza-se uma função de pertinência *triangular*, de menor complexidade, a fim de reduzir o tempo computacional do processo de *Defuzzyficação*.

A aplicação do Sistema Fuzzy viabiliza a avaliação, com maior sensibilidade, das características da rota, pois a aplicação das regras de base permite lidar com a não linearidade, consequência natural da avaliação considerando mais de um fator simultaneamente. Portanto, a justificativa da aplicação do Sistema Fuzzy está relacionada com a possibilidade de moldar as regras de base, permitindo aplicação mais direta de conhecimentos obtidos em estudos prévios.Na seção seguinte, será feita a avaliação de desempenho do algoritmo proposto, analisado comparativamente com DJK, MMRDS e a implementação *Fuzzy* [Ribeiro 2014].

4. Avaliação de Desempenho

A Figura 10 apresenta as topologias USA e NSFNet, utilizadas para realização das simulações. As características e parâmetros utilizados em cada topologia são apresentados na Tabela 2. A Tabela 3 apresenta as parametrizações comuns as duas topologias.

Para realização das simulações foi utilizada a ferramenta SNetS (*Slice Network Simulator*) [de Sousa Santos 2015]. O SNetS permite a avaliação de desempenho em Redes Ópticas Elásticas sob um amplo conjunto de métricas. A simulação de uma topologia se dá em sua fase operacional e considera o comportamento dinâmico de cenários reais. Além disso, a ferramenta fornece meios para a implementação de novas técnicas de roteamento e alocação de espectro, facilitando o processo de elaboração de novas heurísticas. Informações sobre validação podem ser encontradas em [de Sousa Santos 2015].

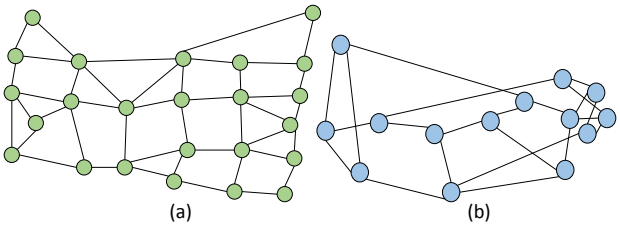


Figura 10. Topologias utilizadas para avaliação: USA(a) e NSFNet(b).

Tabela 2. PARÂMETROS DE TOPOLOGIAS UTILIZADAS NO ESTUDO.

Parâmetros	USA	NSFNet
Número de Nós	24	14
Número de Enlaces	43	22
Carga Inicial por Nó	2.1	3.0
Incremento da Carga	0.36	0.24

Tabela 3. PARÂMETROS DE SIMULAÇÃO.

Parâmetros	
Número de Requisições	100,000
Quantidade de Replicações	10
Pontos de Carga	7
Slots por Enlace	400
Intervalo de Frequência (slot)	12.5 GHz
Largura de Banda das Requisições	10, 20, 40, 80, 160, 320 Gbps

Todos os enlaces nas topologias avaliadas são bidirecionais. A carga de tráfego é distribuída de modo uniforme entre todos os $pares(o, d)$. As requisições são geradas seguindo um processo de Poisson com média λ e o tempo de retenção é distribuído exponencialmente com média $1/\mu$. A intensidade do tráfego na rede é dado por $\rho = \lambda/\mu$. Para alocação de espectro, é utilizada a política $FirstFit$.

A técnica de roteamento por lógica *Fuzzy* permite escolher a rota com menor valor V. Dentre as rotas candidatas para um dado par, a rota escolhida será aquela de melhor avaliação após a *Defuzzyficação*, resultado de menores valores para as métricas Fragmentação Relativa, Índice do Primeiro Slot Alocável e Quantidade de Slots Ocupados. Um baixo valor para esses três critérios resulta em um baixo valor para V.

A Figura 11 apresenta os resultados gráficos de Probabilidade de Bloqueio de Circuito para as topologias (a) USA e (b) NSFNet. O algoritmo proposto apresenta melhora de desempenho devido à sua capacidade de escolher rotas menos congestionadas

e com o espectro menos fragmentado. Assim, ao longo da fase operacional da rede, o espectro encontra-se mais organizado durante um maior período de tempo, permitindo o estabelecimento de mais circuitos. Os outros algoritmos não operam de forma ciente de Fragmentação, e apresentam desempenho inferior, alcançando taxas de bloqueio maiores (uma média de 2.76 vezes maior para o Dijkstra, 2.31 para a proposta *Fuzzy* [Ribeiro 2014] e 1.56 para o MMRDS na topologia USA, 31.82 para o Dijkstra, 6.12 para a outra proposta *Fuzzy* [Ribeiro 2014] e 4.04 para o MMRDS na topologia NSFNet, calculada a partir da média encontrada para cada ponto de carga do gráfico).

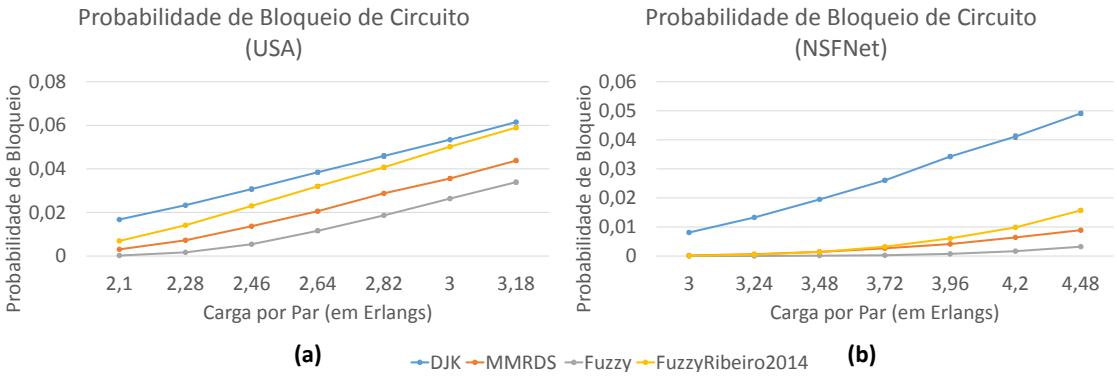


Figura 11. Probabilidade de Bloqueio de Circuito para as topologias: (a) USA e (b) NSFNet.

Durante as simulações também foram avaliadas as taxas de bloqueio para os diferentes tipos de requisição, referente à necessidade de largura de banda de cada uma delas. Para o experimento, foram implementadas requisições que necessitavam de taxas de transmissão iguais a 10, 20, 40, 80, 160 e 320 Gbps, com distribuição uniforme. Por necessitar de grande quantidade de slots, requisições com maiores taxas tem maior probabilidade de sofrer bloqueio. Como consequência, foi observada uma diferença notável entre a Probabilidade de Bloqueio das taxas de maior e menor necessidade de largura de banda. A Figura 12 apresenta a Probabilidade de Bloqueio de Banda para as topologias (a) USA e (b) NSFNet. Os gráficos foram criados a partir do valor de probabilidade de bloqueio por banda observado no último ponto de carga do gráfico da Fig. 11. As taxas de 10 Gbps e 20 Gbps apresentavam valores de bloqueio extremamente baixo em relação aos outros, e foram removidos para proporcionar melhor visualização.

Os valores de Probabilidade de Bloqueio por Banda são componentes da Probabilidade de Bloqueio de Circuito, e pode-se observar o reflexo do comportamento encontrado nos gráficos da Figura 11 quando comparados aos gráficos da Figura 12. Para a topologia NSFNet, observa-se para Probabilidade de Bloqueio de Circuito uma maior proximidade do MMRDS com o *Fuzzy* proposto neste trabalho. Essa mesma proximidade também é observada na probabilidade de Bloqueio por Banda.

Para as topologias USA e NSFNet, a implementação *Fuzzy* proposta apresentou valores de Probabilidade de Bloqueio por Banda inferior aos valores encontrados por DJK, MMRDS e por *Fuzzy* [Ribeiro 2014], considerando as diferentes taxas de larguras de banda avaliadas. Para a topologia USA, a técnica proposta apresenta bloqueio 30,43% inferior ao DJK, 20,87% inferior à proposta *Fuzzy* [Ribeiro 2014] e 13,24% inferior ao MMRDS. Para a topologia NSFNet uma maior variação é observada: a técnica

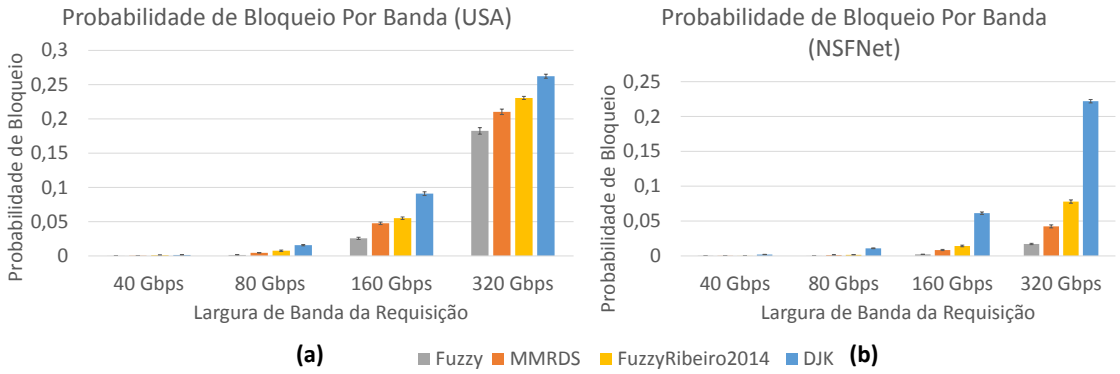


Figura 12. Probabilidade de Bloqueio por Banda para as topologias: (a) USA e (b) NSFNet.

Fuzzy apresenta bloqueio 95,45% inferior ao DJK, 78,03% inferior à proposta *Fuzzy* [Ribeiro 2014] e 63,81% inferior ao MMRDS. Estes valores foram encontrados para as taxas de bloqueio das requisições de largura de banda iguais a 320 Gbps.

A utilização de critérios de avaliação, que verificam o estado de fragmentação e congestionamento das rotas dentro do sistema *Fuzzy*, viabilizou a elaboração de um algoritmo de roteamento ciente de Fragmentação com bom desempenho quando comparado a outros algoritmos de roteamento. Observando os gráficos apresentados, a técnica *Fuzzy* proposta mostra-se eficiente para solucionar o problema de roteamento, pois apresenta menores taxas de bloqueio dos circuitos a serem estabelecidos. A técnica proposta também apresenta menor bloqueio para as várias requisições de diferentes necessidades de largura de banda. Como trabalhos futuros, serão feitas modificações na técnica de roteamento proposta para o desenvolvimento de um algoritmo integrado, que resolve a escolha de rota e a alocação espectral de forma simultânea, a fim de alcançar taxas de bloqueio ainda menores com a aplicação integral do sistema *Fuzzy* no problema RSA.

5. Conclusões

Para suportar a crescente demanda por banda e garantir atendimento a diferentes tipos de circuitos, é necessário uma estrutura de rede que forneça suporte ao crescente tráfego. Assim, surgem as Redes Ópticas Elásticas, que utilizam a luz como vetor de dados, através de diferentes intervalos de frequência, denominados $slots$, que operam de forma independente. Esses intervalos podem ser agrupados para atender requisições que necessitem de maior largura de banda, garantindo flexibilidade para estabelecimento de circuitos com maior necessidade de banda passante.

As Redes Ópticas Elásticas apresentam as *restrições de continuidade e contiguidade espectral*. Juntamente com o contínuo estabelecimento e desligamento de circuitos, essas restrições acabam criando um problema característico das Redes Elásticas, denominado *fragmentação*. Este problema é caracterizado pelo surgimento de pequenos intervalos de slots livres no espectro óptico, de modo que requisições com maiores larguras de banda são incapazes de alocá-los.

A fim de reduzir os impactos da fragmentação, o problema RSA pode ser elaborado considerando os possíveis bloqueios por fragmentação, sendo classificado como algoritmo RSA ciente de fragmentação. O algoritmo proposto neste trabalho é ciente de

Fragmentação e utiliza um sistema *Fuzzy* para realizar roteamento entre os nós da rede. O algoritmo considera um conjunto de métricas, como *Fragmentação Relativa*, *Quantidade de Slots Ocupados* e *Índice do Slot Inicial do Primeiro Intervalo Espectral Alocável*, para escolher a rota a ser alocada entre as rotas de menor caminhos para cada par de nós. Esses critérios permitem a busca de rotas menos fragmentadas e com menor carga, realizando um balanceamento de carga dinamicamente à medida em que surgem novas requisições.

Foram realizadas simulações, nas quais foram avaliados o desempenho do sistema *Fuzzy* proposto comparado aos algoritmos DJK, MMRDS e uma outra proposta *Fuzzy*, em duas topologias reais: USA e NSFNet. Foi constatado que, para o mesmo cenário de estudo, o sistema *Fuzzy* apresenta menor Probabilidade de Bloqueio de Circuito (uma média de 2.76 vezes inferior ao Dijkstra, 2.31 à proposta *Fuzzy* [Ribeiro 2014] e 1.56 ao MMRDS na topologia USA, e 31.82 ao Dijkstra, 6.12 à proposta *Fuzzy* [Ribeiro 2014] e 4.04 ao MMRDS na topologia NSFNet), garantindo o estabelecimento de maior número de circuitos. Portanto, observa-se que a utilização do sistema *Fuzzy* é eficaz para solucionar o problema de roteamento no cenário de Redes Ópticas Elásticas, tornando-se uma alternativa viável para aplicação em cenários reais.

Referências

Alferness, R. (2012). The evolution of configurable wavelength multiplexed optical networks x2014;a historical perspective. *Proceedings of the IEEE*, 100(5):1023–1034.

Bhanja, U. and Mishra, D. (2015). Dynamic routing and wavelength assignment using fuzzy logic controller in wdm optical networks. In *Signal Processing and Communication (ICSC), 2015 International Conference on*, pages 380–385.

Cai, J.-X., Davidson, C., Lucero, A., Zhang, H., Foursa, D., Sinkin, O., Patterson, W., Pilipetskii, A., Mohs, G., and Bergano, N. S. (2012). 20 tbit/s transmission over 6860 km with sub-nyquist channel spacing. *Lightwave Technology, Journal of*, 30(4):651–657.

Cugini, F., Paolucci, F., Meloni, G., Berrettini, G., Secondini, M., Fresi, F., Sambo, N., Poti, L., and Castoldi, P. (2013). Push-pull defragmentation without traffic disruption in flexible grid optical networks. *Lightwave Technology, Journal of*, 31(1):125–133.

de Sousa Santos, I. G. (2015). Alocação de recursos para o estabelecimento de circuitos em redes ópticas wdm e ofdm. *Universidade Federal do Piauí*.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

dos Reis, J., Raddo, T., Sanches, A., and Borges, B.-H. (2015). Fuzzy logic control for the mitigation of environmental temperature variations in ocdma networks. *Optical Communications and Networking, IEEE/OSA Journal of*, 7(5):480–488.

Gerstel, O., Jinno, M., Lord, A., and Yoo, S. (2012). Elastic optical networking: a new dawn for the optical layer? *Communications Magazine, IEEE*, 50(2):s12–s20.

Horota, A. K., Figueiredo, G. B., and da Fonseca, N. L. S. (2014). Algoritmo de roteamento e atribuição de espectro com minimização de fragmentação em redes óticas elásticas. *Anais do 32 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2014*, 32:895–908.

Iancu, I. (2012). *A Mamdani type fuzzy logic controller*. INTECH Open Access Publisher.

Kavian, Y., Rejeb, R., and Strobel, O. (2010). Fuzzy linear programming for capacity planning in optical networks. In *Transparent Optical Networks (ICTON), 2010 12th International Conference on*, pages 1–4.

Kretsis, A., Christodoulopoulos, K., Kokkinos, P., and Varvarigos, E. (2014). Planning and operating flexible optical networks: Algorithmic issues and tools. *Communications Magazine, IEEE*, 52(1):61–69.

Moura, P., Da Fonseca, N., and Scaraficci, R. (2014). Fragmentation aware routing and spectrum assignment algorithm. In *Communications (ICC), 2014 IEEE International Conference on*, pages 1137–1142.

Pappis, C. P. and Siettos, C. I. (2005). Fuzzy reasoning. In *Search Methodologies*, pages 437–474. Springer.

Ribeiro, S. R. L. (2014). Roteamento multicritério em redes ópticas elásticas dinâmicas. *Instituto Federal de Educação, Ciência e Tecnologia da Paraíba*.

Santos, I. G. S., Duraes, G., Giozza, W., Soares, A., and Catu, B. (2012). Um novo algoritmo de roteamento para a escolha da melhor entre as menores rotas. In *SBRC - Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Shakya, S. and Cao, X. (2013). Spectral defragmentation in elastic optical path networks using independent sets. In *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013*, pages 1–3.

Tomkos, I., Mukherjee, B., Korotky, S. K., Tucker, R., and Lunardi, L. (2012). The evolution of optical networking [scanning the issue]. *Proceedings of the IEEE*, 100(5):1017–1022.

Wan, X., Hua, N., and Zheng, X. (2012). Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *Optical Communications and Networking, IEEE/OSA Journal of*, 4(8):603–613.

Wang, R. and Mukherjee, B. (2012). Spectrum management in heterogeneous bandwidth networks. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 2907–2911.

Wang, R. and Mukherjee, B. (2014). Spectrum management in heterogeneous bandwidth optical networks. *Optical Switching and Networking*, 11, Part A:83 – 91.

Wright, P., Parker, M., and Lord, A. (2015). Minimum- and maximum-entropy routing and spectrum assignment for flexgrid elastic optical networking [invited]. *Optical Communications and Networking, IEEE/OSA Journal of*, 7(1):A66–A72.

Yin, Y., Zhang, H., Zhang, M., Xia, M., Zhu, Z., Dahlfort, S., and Yoo, S. (2013). Spectral and spatial 2d fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks [invited]. *Optical Communications and Networking, IEEE/OSA Journal of*, 5(10):A100–A106.

Novo Algoritmo RMLSA com Roteamento *Multihop* em Redes Ópticas Elásticas

Lucas R. Costa, André C. Drummond

[1]Departamento de Ciência da Computação – Universidade de Brasília (UnB), 70910-900, Brasília, Brasil

{lucasrc,andred}@unb.br

Abstract. *Elastic Optical Networks (EON) emerges as a promising solution for the future of high-speed optical transmission. Its features provide flexibility and superior scalability in spectrum allocation for the various emerging services following the increasing demand of Internet traffic. In EON traffic demands are supported by a spectrum allocation, using an arbitrary number of OFDM subcarriers, and a suitable modulation level, taking into account the transmission distance. The purpose of this work is to address the Routing, Modulation Level, and Spectrum Allocation problem, called RMLSA, through the use of electric and optical traffic grooming associated with the control of spectral modulation in dynamics traffic scenario. To this end, we propose an algorithm that seeks to hold the largest possible amount of optical grooming using higher levels of modulation by using multihop in virtual topology. The results show the reduction in blocking ratio without compromising the use of network resources.*

Resumo. *A tecnologia de redes ópticas elásticas surge como uma solução promissora para o futuro da transmissão óptica de alta velocidade. Suas características proporcionam uma flexibilidade e escalabilidade superior na alocação de espectro para os diversos serviços emergentes acompanhando a crescente demanda do tráfego da Internet. Em redes ópticas elásticas as demandas de tráfego são suportadas por uma atribuição de espectro, usando um número arbitrário de subportadoras OFDM, e um nível de modulação adequado, tendo em conta a distância de transmissão. A proposta deste trabalho, procura resolver o problema de roteamento e atribuição de espectro com modulação adaptativa, chamado RMLSA (Routing, Modulation Level, and Spectrum Allocation), através do uso da agregação de tráfego associado ao controle da modulação espectral no cenário de tráfego dinâmico. Para este fim, propõe-se um algoritmo que procura realizar a maior quantidade de agregação óptica possível usando os níveis de modulação mais altos através de múltiplos saltos na topologia virtual. Os resultados mostram a redução na taxa de bloqueio sem comprometer a utilização dos recursos na rede.*

1. Introdução

O tráfego na Internet tem crescido exponencialmente nos últimos anos e as previsões indicam que este crescimento continuará devido as aplicações emergentes, tais como, aplicações multimídia, TV de alta definição, computação em nuvem e aplicações de rede em tempo real. A tecnologia de transmissão óptica convencional é insuficiente para atender

as crescentes demandas de tráfego da Internet, uma vez que a mesma sofre com limitações físicas que impõem a transmissão de taxas fixas em cada comprimento de onda, reduzindo a eficiência na utilização dos seus recursos [Saradhi and Subramaniam 2009]. A adaptação da tecnologia de transmissão óptica convencional para as novas demandas de tráfego é um dos desafios para a Internet do Futuro. Para tratar adequadamente esse desafio são necessárias redes flexíveis equipadas com elementos que possam se adaptar às demandas heterogêneas do tráfego.

O paradigma de redes ópticas elásticas (*Elastic Optical Networking* - EON) surgiu com esse objetivo. Uma rede EON tem a capacidade de ajustar dinamicamente seus recursos, tais como, a largura de banda óptica e o formato de modulação de acordo com os requisitos de cada demanda. Para alocação de recursos de espectro em uma rede EON é necessário encontrar uma rota e atribuir à mesma uma quantidade contígua de *slots* espectrais [Tomkos et al. 2014], este problema é chamado RSA (*Routing and Spectrum Assignment*). Recentemente esse problema evoluiu para o chamado RMLSA (*Routing, Modulation Level, and Spectrum Allocation*) [Christodoulopoulos et al. 2011] que adiciona a atribuição do formato de modulação ao espectro na fibra. Diversos algoritmos para a alocação de recursos em redes EON têm sido propostos na literatura. Tais estudos demonstram a eficácia do novo paradigma e sua viabilidade.

Este trabalho propõe uma técnica que resolve o problema RMLSA através do uso da agregação elétrica e óptica associado ao controle da modulação espectral no cenário de tráfego dinâmico. O objetivo de associar essas técnicas visa aproveitar melhor os recursos da rede óptica atendendo maiores taxas de dados. Para este fim, propõe-se um algoritmo que procura realizar a maior agregação óptica possível usando os níveis de modulação mais altos através de múltiplos saltos na topologia virtual. Os resultados mostram uma redução na taxa de bloqueio de até 50%, proporcionando ganhos significativos sem comprometer a utilização dos recursos na rede.

O restante do artigo está organizado como segue. A Seção 2 apresenta brevemente a arquitetura das redes ópticas elásticas, bem com os elementos necessários para seu funcionamento. A Seção 3 apresenta o estado da arte das redes ópticas elásticas, algoritmos e técnicas propostas na literatura. A Seção 4 apresenta a técnica proposta neste trabalho para resolver o problema RMLSA. A Seção 5 mostra os resultados numéricos com o uso da técnica proposta. Por fim a Seção 6 apresenta as considerações finais.

2. Redes Ópticas Elásticas

As redes EON possuem a característica de dividir os recursos espectrais em *slots* de frequência na forma de subportadoras, permitindo múltiplos formatos de modulação e taxas de dados e espectro de tamanhos variados. Nesse contexto, o objetivo de uma EON é alocar uma demanda a um determinado caminho óptico que possua uma largura de banda óptica com um tamanho apropriado à mesma. A Figura 1(a) apresenta as diferenças entre os caminhos ópticos com grade fixa e flexível. Na grade fixa a faixa de frequência do espectro utilizado é única independentemente da demanda requisitada pelo cliente. Na grade flexível essa faixa de frequência é adaptável de acordo com a demanda requisitada. Assim, um caminho óptico elástico pode transmitir múltiplas taxas de dados, pois seu tamanho é alocado de acordo com a demanda requisitada pelo cliente. A arquitetura da rede EON é composta por transmissores de largura de banda variável (*bandwidth-variable*

transponder - BVT) e por comutadores ópticos de banda variável (*bandwidth-variable Wavelength Cross-Connects* - BV-WXC) que permitem o estabelecimento de caminhos ópticos com grade flexível. A Figura 1(b) mostra a arquitetura de uma rede EON.

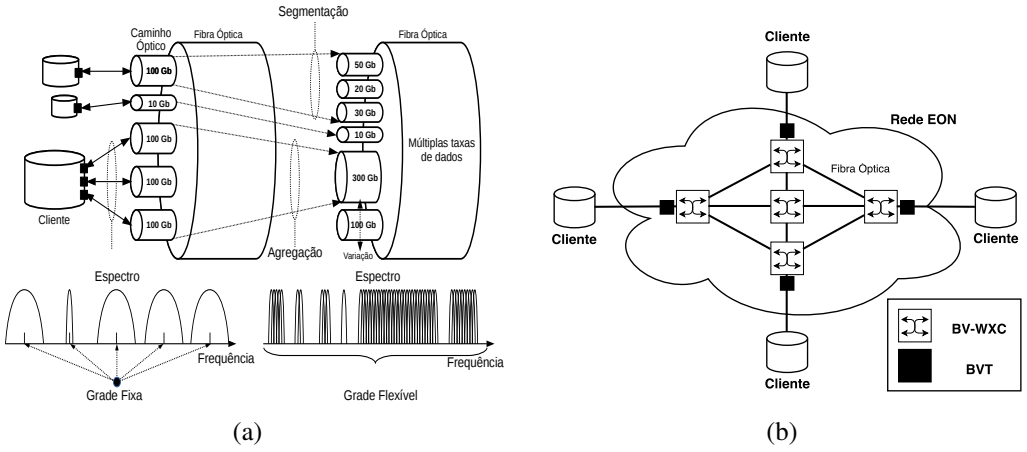


Figura 1. (a) Comparação entre os caminhos ópticos de redes com grade fixa e flexível. (b) Arquitetura das redes EON.

Os BVTs são responsáveis por alocar espectro suficiente para acomodar cada demanda. Várias subportadoras OFDM podem ser fundidas para formar um supercanal transportando os dados sem bandas de guarda no espectro. Dessa forma, os BVTs criam caminhos ópticos com largura de banda flexível permitindo o ajuste dos recursos ópticos de acordo com a demanda necessária [Zhang et al. 2013]. Os BV-WXC são os elementos responsáveis por estabelecer um caminho óptico fim-a-fim com a largura de banda necessária para acomodar os recursos espectrais estabelecidos pelos BVTs. Quando os BVTs aumentam a taxa do tráfego, cada BV-WXC na rota deve expandir sua janela de comutação, permitindo assim uma taxa de dados variável em cada caminho óptico [Zhang et al. 2013].

Nas redes EON, o formato de modulação é outro aspecto que permite o ajuste flexível da largura de banda. Em particular, cada caminho óptico, pode ser modulado individualmente utilizando um modulador diferente para cada transmissão. O número de subportadoras e o formato de modulação utilizado é ajustável de acordo com a quantidade de tráfego e o alcance óptico requisitado [Zhang et al. 2013]. A escolha do nível de modulação deve levar em consideração a qualidade necessária de transmissão (QoT) e, consequentemente, a tolerância de relação sinal-ruído óptico [Zhang et al. 2013, Christodoulopoulos et al. 2011]. Uma abordagem comumente utilizada pela literatura EON define a distância de transmissão do caminho óptico como o fator mais relevante na QoT [Jinno et al. 2010]. Portanto, a escolha do formato de modulação a ser utilizado é estabelecida de acordo com a distância do caminho percorrido na fibra. Dessa maneira, transmissões através de caminhos ópticos mais curtos são capazes de utilizar níveis de modulação mais altos.

A banda passante de cada subportadora está, portanto, intimamente relacionada à largura do espectro de cada subportadora e sua modulação adotada. A Equação 1 apresenta essa relação [Xiong 2006]:

$$B = \frac{C}{log_2 M} \tag{1}$$

onde B é a largura de banda do espectro da subportadora em GHz, C é a taxa de dados em Gbps e M é o nível de modulação M-QAM ou M-PSK adotado. Em outras palavras, *M* é o número de fases usadas para codificar um determinado número de *bits* por símbolo. Portanto, quanto maior o nível de modulação, maior será a banda passante da subportadora e menor será seu alcance de acordo com o requisito de QoT.

Para estabelecer um caminho óptico com largura de espectro flexível é necessário resolver o problema RSA. O problema RSA exige que as subportadoras, pertencentes ao mesmo caminho óptico, sejam roteadas de forma contígua utilizando a mesma faixa de espectro durante toda a rota. Caminhos ópticos distintos devem ser separados por uma banda de guarda para atender as restrições da tecnologia OFDM. Semelhante à tecnologia de agregação elétrica das redes convencionais, em redes EON também é possível agregar fluxos a um caminho óptico já estabelecido [Zhang et al. 2011]. A tecnologia OFDM também permite outro tipo de agregação do tráfego, a agregação através do meio óptico. A agregação óptica permite aproveitar a flexibilidade proporcionada pelos comutadores EON agregando múltiplos caminhos ópticos em um único transmissor e comutá-los opticamente em conjunto [Zhang et al. 2012]. Esse agrupamento é realizado para suportar agregação de modo transparente, sem conversão do sinal do domínio óptico para elétrico. Esse grupo de caminhos ópticos é chamado de túnel óptico.

3. Trabalhos Relacionados

Em [Wan et al. 2011], os autores propuseram três heurísticas para solucionar o problema RSA em um cenário dinâmico. A primeira heurística (KSP) é um algoritmo dividido em dois passos, em que é utilizado o algoritmo Yen‘s KSP para encontrar as k-menores rotas da origem ao destino da demanda e em seguida o algoritmo tenta alocar a demanda em uma dessas k-rotas. A segunda heurística apresentada foi o algoritmo MSP (*Modified Dijkstra Shortest Path*), que procura a menor rota possível para alocar a demanda de tráfego baseado no algoritmo de *Dijkstra*. A terceira heurística foi o algoritmo SPV (*Spectrum-Constraint Path Vector Searching*), que consiste em um algoritmo de busca e construção de uma árvore para encontrar o menor caminho com espectro disponível para atender à demanda. Os resultados apresentados demonstram uma redução significativa na taxa de bloqueio em relação a algoritmos de RWA tradicionais em redes WDM.

O problema RMLSA foi inicialmente apresentado em [Christodoulopoulos et al. 2011], onde os autores adicionam o formato de modulação ao problema RSA. Posteriormente, o problema RMLSA foi investigado em [Wan et al. 2012], onde os autores avaliam os efeitos da modulação adaptativa nos mesmos algoritmos de [Wan et al. 2011]. Os autores propõe um esquema de modulação adaptativa para o problema RSA denominado *mAdap*. O esquema pode ser visto na Figura2. A cada chegada de fluxo o algoritmo escolhe a maior modulação disponível na rede, em seguida calcula o número de *slots* necessários, dada esta modulação e o tamanho em *Hz* dos *slots*, para atender à demanda. Em seguida calcula-se a rota, de acordo com algum algoritmo de RSA e verifica se existe uma rota disponível que atenda os requisitos de continuidade e contiguidade do problema RSA. Caso não exista rota

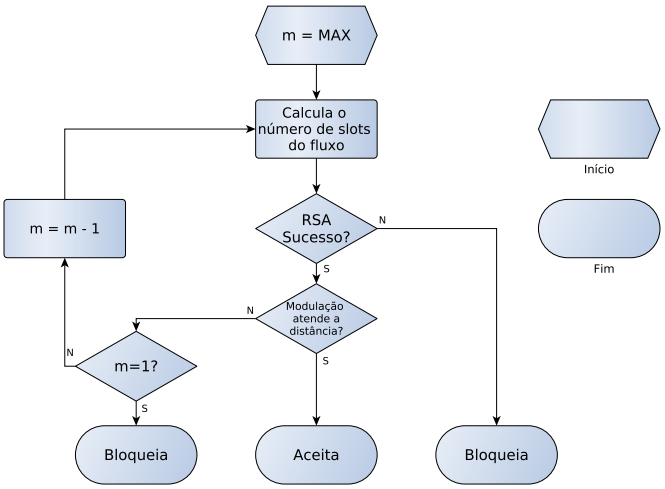


Figura 2. Esquema de modulação adaptativa para o problema RSA *mAdap*.

disponível, bloqueia-se a requisição, caso contrário, verifica se a distância percorrida nesta rota atende os requisitos mínimos de QoT da modulação. Caso a rota atenda os requisitos mínimos de QoT da modulação escolhida, a requisição é alocada na rede e aceita, caso não atenda, o nível de modulação é diminuído e recalcula-se o número de *slots* necessários para atender à demanda neste novo nível de modulação. Esse laço é executado até o menor nível de modulação possível da rede. Se mesmo no menor nível de modulação possível, o algoritmo RSA fornecer uma rota que não atende os requisitos de QoT da modulação então a requisição também é bloqueada. Os resultados apresentados dessa nova abordagem são comparados com uma PLI (Programação Linear Inteira) proposta e apresentam uma redução significativa na taxa de bloqueio e no uso do espectro [Wan et al. 2012].

Nas redes ópticas convencionais o uso de agregação de tráfego é comumente utilizado para elevar a usabilidade da rede óptica. Nas redes EON essa usabilidade pode ser ainda maior devido à tecnologia de agregação óptica. Os autores em [Khodashenas et al. 2013] propõem o algoritmo FPA (*First-Possible Aggregating*) que realiza agregação óptica no cenário de rede dinâmico. O algoritmo FPA maximiza a utilização da capacidade do BVT agregando múltiplos caminhos ópticos no mesmo transmissor. Além disso, o algoritmo FPA melhora a utilização do espectro através da redução do número de bandas de guarda entre os caminhos ópticos que compartilham a mesma rota. Esses recursos podem ser facilmente transformados para mais conexões estabelecidas através da rede e um menor número de transmissores exigidos por nó. Os resultados demonstraram que o uso da agregação óptica traz benefícios significativos na taxa de bloqueio de requisições em relação a algoritmos que não utilizam a técnica.

Recentemente em [Zhang et al. 2015], os autores propõem o uso de um grafo auxiliar com a implementação de técnicas de agregação elétrica e óptica para a resolução do problema RSA. Os autores também propõem dois esquemas de reserva de espectro para a agregação de futuras demandas de tráfego. São comparadas diferentes políticas de agregação em termos da quantidade de banda bloqueada, número de transmissores, número de saltos na topologia virtual e física. Os resultados da simulação mostraram que há

uma compensação entre as políticas e ambas devem ser implementadas de acordo com os objetivos do operador de rede.

Embora os algoritmos propostos em [Wan et al. 2012] levem em consideração a modulação adaptativa, recursos como agregação elétrica e óptica são ignorados pelos algoritmos. Os algoritmos propostos em [Khodashenas et al. 2013] e [Zhang et al. 2015] embora levem em consideração técnicas de agregação de tráfego, não abordam o contexto de modulação adaptativa nas suas propostas. Os autores em [Ye et al. 2014] propõem o primeiro algoritmo de distância adaptativa com técnicas de agregação de tráfego. No algoritmo DFG (*Distance-adaptive and Fragmentation-aware optical Grooming algorithm*) proposto, as demandas de tráfego com mesma origem que compartilham o mesmo enlace são agregadas opticamente e comutadas pelo meio óptico sem a utilização de bandas de guarda. O algoritmo ainda leva em consideração a modulação utilizada pelo túnel óptico e mostra que nem sempre a utilização da agregação óptica é eficiente, pois a distância de um determinado caminho óptico dentro do túnel pode ser muito longa e prejudicar a modulação do túnel óptico como um todo, como pode ser visto na Figura 3. Na Figura 3(a) o algoritmo FPA atende as quatro demandas com uso de 14 *slots* mais 2 bandas de guarda, um total de 16 *slots*, na Figura 3(b) (algoritmo DFG) as mesmas demandas são atendidas com um total de 14 *slots*, pois foi utilizado outro nível de modulação nos nós *d1*, *d2* e *d3*.

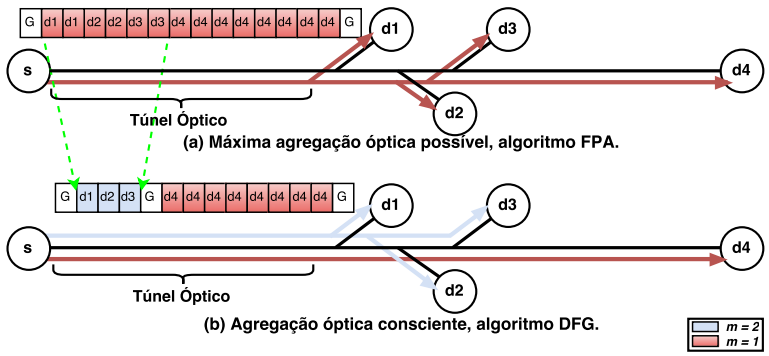


Figura 3. Problema de distância adaptativa com agregação óptica.

Embora o algoritmo em [Ye et al. 2014] leve em consideração o uso de técnicas de agregação e modulação adaptativa, o trabalho não leva em consideração o ambiente de tráfego de rede dinâmico, somente a abordagem estática. Outro fator importante não investigado é o uso de múltiplos saltos na topologia virtual (roteamento *multihop*). A Tabela 1 sumariza as propostas da literatura e apresenta suas características.

Tabela 1. Sumário de pesquisas sobre abordagens RSA.

Proposta	Tráfego	Agregação de Dados	Agregação de Espectro	Roteamento	Modulação
KSP, MSP, SPV [Wan et al. 2011]	Dinâmico	Não	Não	Single-Hop	Estática
RMLSA [Christodoulopoulos et al. 2011]	Estático	Não	Não	Single-Hop	Dinâmica
FPA [Khodashenas et al. 2013]	Dinâmico	Não	Sim	Single-Hop	Estática
MPH-SRNP [Zhang et al. 2015]	Dinâmico	Sim	Sim	Multi-Hop	Estática
DFG [Ye et al. 2014]	Estático	Sim	Sim	Single-Hop	Dinâmica
MBM (nossa proposta)	Dinâmico	Sim	Sim	Multi-Hop	Dinâmica

4. Proposta de Trabalho

Este artigo propõe um algoritmo que promove o uso do maior nível de modulação suportado pela rede. O algoritmo MBM (*Maximize the use of Best Modulation format*) proposto procura realizar o maior uso da agregação óptica possível associado ao uso do nível de modulação mais alto. Dessa forma, para que uma transmissão óptica entre dois nós distantes seja realizada com o uso de um nível de modulação alto é necessário que essa transmissão óptica seja composta por vários caminhos ópticos mais curtos que possam atender os requisitos de QoT dos níveis de modulação mais altos. A Figura 4 mostra um exemplo da sua lógica de funcionamento. Seguindo o mesmo exemplo da Figura 3, o nó *d4* está muito distante do nó de origem *s* e por sua vez é alcançado com QoT aceitável somente com o nível de modulação $m = 1$. Por outro lado, os nós de destino *d1*, *d2* e *d3* podem ser alcançados pelo nível de modulação $m = 2$. A proposta de [Ye et al. 2014], Figura 3(b), observa essa situação e divide o túnel óptico em dois túneis ganhando recursos de espectro. No entanto, a demanda que contém a maior distância continua com o mesmo nível de modulação. No caso do algoritmo MBM, o uso do *multihop* traz a vantagem de usar caminhos ópticos menores e, por sua vez, utilizar níveis de modulação maiores.

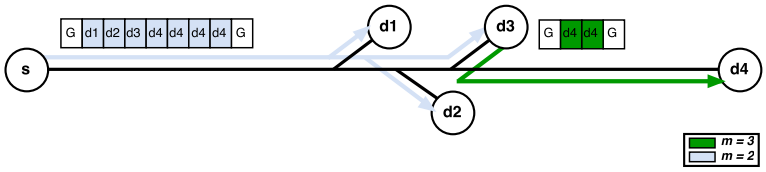


Figura 4. Exemplificação do algoritmo MBM.

O exemplo da Figura 4 mostra a vantagem do uso do algoritmo MBM. Como no algoritmo DFG, as demandas com destino aos nós *d1*, *d2* e *d3* continuam sendo enviadas pelo nível de modulação $m = 2$, entretanto a demanda com destino ao nó *d4*, que requeria 8 *slots* no nível de modulação $m = 1$, é enviada até o nó *d3* com o nível de modulação $m = 2$, através do uso da agregação óptica. Quando a demanda com destino *d4* chega ao nó *d3* ocorre uma conversão OEO e outro transmissor é utilizado com um nível de modulação maior possível, que neste caso é $m = 3$. Dessa forma, o algoritmo MBM utiliza ao todo 13 *slots* para enviar as quatro demandas enquanto o algoritmo DFG utiliza 14 *slots*. O algoritmo MBM foi projetado para potencializar o esquema de modulação adaptativa *mAdap* proposto em [Wan et al. 2012]. Todavia, o MBM não requer a aplicação do esquema *mAdap* para resolver o problema RMLSA.

A abordagem utilizada pelo algoritmo MBM é descrita no Algoritmo 1, onde *AE* representa a agregação elétrica e *AO* representa a agregação óptica. Inicialmente cria-se duas estruturas compostas por k-menores caminhos disjuntos e não disjuntos através de uma execução *off-line* (linhas 1-2). Em seguida, para cada requisição, o algoritmo MBM procura um caminho óptico alocado na rede para realizar uma *AE* através da política *least loaded* (linha 4). Caso não seja possível realizar *AE* o algoritmo obtém os k menores caminhos disjuntos de s até d (linha 7) e tenta realizar uma *AO* procurando por caminhos ópticos com origem s que compartilham o máximo de enlaces possíveis com $Paths[i]$ (linhas 8-10). Caso não encontre um caminho que atenda esses requisitos, parte-se para a estratégia *multihop*. O algoritmo obtém os k-menores caminhos (linha 15) e procura dentre os mesmos se é possível realizar *AE* ou *AO* sobre os caminhos ópticos que compõem o

Algoritmo 1 *Maximize the use of Best Modulation format* − MBM

```
1: kDPaths ← todos os k-menores caminhos disjuntos (off-line)
2: kSPaths ← todos os k-menores caminhos (off-line)
3: para cada requisição de conexão r(s, d, b) faça
4:     se possível fazer AE com r de s até d então                    ▷ (Estratégia singlehop)
5:         aceita a requisição r
6:     senão
7:         Paths ← kDPaths(s, d)
8:         para i ← 1 até k faça
9:             Procura um caminho óptico com origem em s que compartilha o máximo de enlaces com Paths[i]
10:            se encontrou um caminho óptico e for possível fazer AO com r de s até d então
11:                aceita a requisição r
12:            fim se
13:        fim para
14:    fim se
15:    Paths ← kSPaths(s, d)                                          ▷ (Estratégia multihop)
16:    para i ← 1 até k faça
17:        s′ ← s
18:        d′ ← s
19:        enquanto d′ ≠ d faça
20:            Procura um caminho óptico com origem em s′ que compartilha o máximo de enlaces com Paths[i]
21:            se encontrou um caminho óptico então
22:                d′ ← o último nó compartilhado
23:                se possível fazer AE com r de s′ até d′ então
24:                    s′ ← d′
25:                senão
26:                    se possível fazer AO com r de s′ até d′ então
27:                        s′ ← d′
28:                    senão
29:                        Sai do laço
30:                    fim se
31:                fim se
32:            senão
33:                Sai do laço
34:            fim se
35:        fim enquanto
36:        se s′ = d então
37:            aceita a requisição r
38:        senão
39:            Criar um novo caminho óptico de s′ até d com o algoritmo KSP com nível de modulação adequado à distância
40:            se for possível criar caminho óptico então
41:                aceita a requisição r
42:            fim se
43:        fim se
44:    fim para                                                       ▷ (Estratégia singlehop)
45:    Executar o algoritmo KSP na requisição r com origem s e destino d com o maior nível de modulação adequado à distância
46:    se for possível criar um caminho óptico então
47:        aceita a requisição r
48:    senão
49:        bloqueia a requisição r
50:    fim se
51: fim para
```

caminho $Paths[i]$, realizando assim, múltiplos saltos na topologia virtual (linhas 16-35). Ainda no final do laço, o algoritmo procura criar um novo caminho óptico com o maior nível de modulação adequado à distância para compor o restante do caminho $Paths[i]$ que não foi possível ser completado pela agregação de tráfego (linhas 36-44). Por fim, se não for possível realizar nenhum tipo de agregação, então o algoritmo executa a estratégia utilizada pelo algoritmo KSP com o nível de modulação adequado à distância usando uma abordagem *singlehop* (linha 45). Se mesmo assim não for possível criar um caminho óptico, então a requisição é bloqueada (linha 49).

É interessante notar, que o uso da estratégia *multihop* do MBM associado ao esquema de modulação adaptativa *mAdap* procura sempre utilizar o maior nível de modu-

lação possível, neste caso, o melhor nível de modulação possível dado a distância. Em outras palavras, o algoritmo MBM potencializa o esquema de modulação adaptativa *mAdap* provocando a utilização dos maiores níveis de modulação suportados na rede, com o intuito de economizar espectro. O uso de múltiplos saltos na topologia virtual provoca a utilização de mais transmissores na rede. Entretanto, a agregação óptica faz com que o aumento do uso de transmissores não seja um fator significativo, pois à medida que novos transmissores são ativados em pontos distribuídos na rede, os mesmos podem ser reaproveitados para a agregação de novas demandas.

A complexidade de tempo do algoritmo MBM é analisada como segue. A primeira parte (linhas 4-14) é dominada por duas etapas: (*i*) a verificação inicial de possível *AE* (linha 4); e (*ii*) a verificação de possível *AO* em k rotas (linhas 8-13). Tendo em vista que o número máximo de caminhos ópticos alocados na rede é $|E||S|$, onde E é o conjunto de enlaces na rede e S é o conjunto de *slots* em cada enlace, tem-se que a complexidade da primeira etapa é $O(|E||S|)$. A complexidade da segunda etapa segue o mesmo raciocínio em k rotas, logo tem-se uma complexidade de $O(k * |E||S|)$. Portanto, a primeira parte tem uma complexidade de tempo equivalente a $O(k * |E||S|)$. A segunda parte (linhas 15-44) é dominada por duas etapas: (*i*) a verificação de possível *AE* (linha 23) e/ou possível *AO* (linha 26); e (*ii*) a execução do algoritmo KSP (linha 39). Seguindo o raciocínio anterior, a complexidade da primeira etapa é $O(|E||S|)$ para realizar *AE* ou *AO*. Na segunda etapa, marcada pela execução do algoritmo KSP, tem-se uma complexidade de $O(k)$, uma vez que o algoritmo YenKSP é executado *offline* [Wan et al. 2012]. Portanto, considerando os laços das linhas (16/19), a segunda parte tem uma complexidade equivalente a $O(k * N * |E||S| + k^2)$, onde N é o número de nós na rede. A terceira parte (linhas 45-50) tem apenas uma etapa que é a execução do algoritmo KSP, logo sua complexidade é de $O(k)$. Como a segunda parte é a que impõem a maior complexidade, tem-se que a complexidade do algoritmo MBM é de $O(k * N * |E||S| + k^2)$ que é polinomial.

5. Resultados Numéricos

As simulações foram realizadas utilizando o simulador WDMSim [Drummond 2016] com adaptações para cenário EON. Cada simulação foi realizada cinco vezes utilizando o método de replicações independentes. Para os resultados apresentados foram calculados intervalos de confiança com 95% de confiabilidade. Em cada simulação foram geradas 10^5 requisições de conexão com 15 níveis de granularidade variando de $12,5$ Gb/s até 100 Gb/s com passos de $6,25$ Gb/s com mesma probabilidade de chegada. O processo de chegada das chamadas segue a distribuição de *Poisson* com origem e destino distribuídos uniformemente para todos os pares de comunicação da rede. As topologias consideradas nas simulações foram a USANet com 24 nós e 43 enlaces bidirecionais, e a topologia PanEuro com 27 nós e 81 enlaces bidirecional. A Figura 5 apresenta as distâncias dos enlaces em quilômetros. A largura de cada *slot* considerada foi $12,5$ GHz e foi assumido que cada enlace possui a capacidade de 120 *slots* ($1,5$ THz). Assume-se uma banda de guarda de 2 *slots* (25 GHz). Cada nó na topologia possui 15 transmissores e cada transmissor tem a capacidade máxima de transmitir até 8 *slots*. As modulações consideradas são BPSK, QPSK, 8QAM e 16QAM com 1, 2, 3 e 4 bits por símbolo e alcances de 8000, 4000, 2000 e 1000 km, respectivamente. Para todos os algoritmos, foi considerado o parâmetro $k = 3$ e a política de FF (*First Fit*) para resolver o problema RSA.

Foram comparados cinco algoritmos da literatura (KSP, MSP, SPV, FPA e MPH-

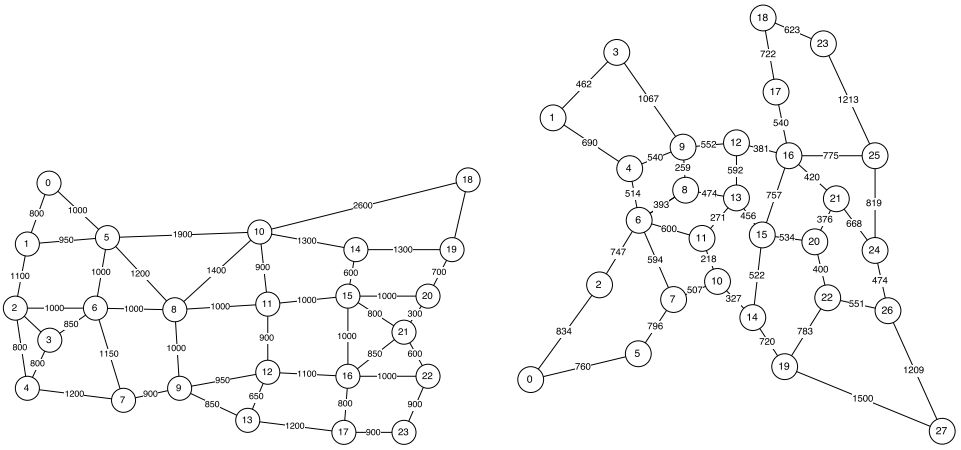


Figura 5. Topologia USANet e PanEuro.

SRNP) com o algoritmo MBM. Os algoritmos KSP, MSP e SPV [Wan et al. 2011, Wan et al. 2012] não consideram técnicas de agregação de tráfego e utilizam a estratégia *singlehop*. O algoritmo FPA [Khodashenas et al. 2013], por sua vez já considera o uso das técnicas de agregação de tráfego mas continua utilizando a estratégia *singlehop*. O algoritmo MPH-SRNP [Zhang et al. 2015] considera o uso das técnicas de agregação de tráfego e utiliza a estratégia *multihop*. Para tornar todas as abordagens avaliadas capazes de resolver o problema RMLSA e garantir um ambiente de avaliação justo, todos os algoritmos foram avaliados sob o esquema *mAdap* (Figura 2) conforme em [Wan et al. 2012].

As seguintes métricas são avaliadas: Taxa de bloqueio de banda (BBR), taxa média de transmissores disponíveis, média de saltos na topologia virtual por requisição, média da taxa de espectro disponível e taxa média do uso de modulação. As métricas referentes à utilização de recursos na rede são avaliadas para evidenciar os ganhos relativos aos mecanismos implementados no algoritmo MBM. Além disso, soluções de engenharia de tráfego que levam à redução na utilização dos recursos instalados na rede, contribuem para a diminuição de sua probabilidade de exaustão.

Taxa de bloqueio de banda (BBR)

O BBR reflete a taxa de banda bloqueada, quanto maior essa taxa, maior é a banda bloqueada. A Figura 6 mostra o BBR dos algoritmos considerados para várias cargas de tráfego nas topologias UsaNet e PanEuro, respectivamente. Pode-se observar que o algoritmo MBM obteve o menor bloqueio de banda frente aos demais algoritmos. Em relação aos algoritmos KSP, MSP e SPV a diferença na taxa de bloqueio caiu pela metade, para os algoritmos FPA e MPH-SRNP a redução foi de até 50% para cargas baixas e até 10% para cargas mais altas. Isso acontece, pois ao considerar múltiplos saltos na topologia virtual o algoritmo tem mais possibilidades para atender as demandas, tendo em vista que os requisitos de continuidade são aliviados. Embora o algoritmo MPH-SRNP também realize *multihop*, sua técnica de minimizar o número de saltos na topologia física faz com que suas rotas percorram maiores distâncias e consequentemente leva o esquema de modulação adaptativa (*mAdap*) a utilizar níveis de modulação menores. O não aproveitamento adequado dos maiores níveis de modulação provoca uma queda no desempenho do MPH-SRNP.

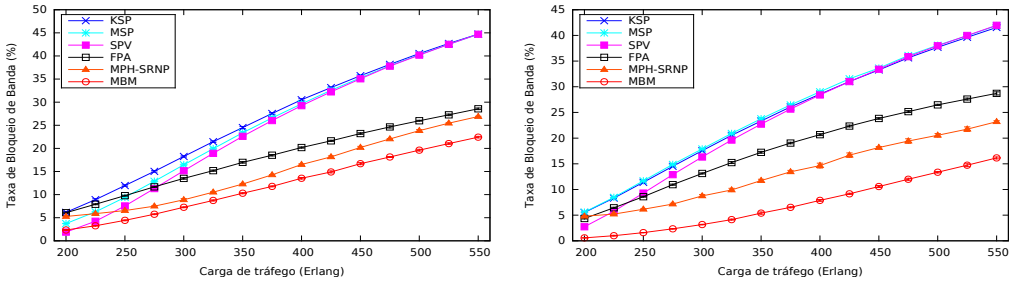


Figura 6. Taxa de bloqueio de banda (BBR). Topologias USANet e PanEuro, respectivamente

Taxa média de transmissores disponíveis

A taxa de transmissores disponíveis reflete a usabilidade do número de transmissores, quanto maior for essa taxa, maior a quantidade de transmissores disponíveis na rede. A Figura 7 mostra os resultados para as topologias UsaNet e PanEuro, respectivamente, tento em vista que cada nó possui 15 transmissores nesta simulação. Observa-se que o uso de transmissores no algoritmo MBM é um pouco superior ao do algoritmo FPA e bem inferior aos demais. No entanto, isso já era esperado, tendo em vista que o uso da técnica de *multihop* exige um maior uso de transmissores na rede. É importante notar a proximidade do MBM com o FPA, cerca de 2% na topologia UsaNet e 5% na topologia PanEuro. Nos demais algoritmos a taxa de utilização de transmissores é cerca de 12% maior que o MBM. Isto demonstra que o aumento do uso de transmissores do algoritmo MBM não trouxe um impacto significativo quando comparado aos ganhos de BBR, o mesmo não acontece ao MPH-SRNP, que utiliza um maior número de transmissores e obtém um baixo ganho de BBR. Isto evidencia a eficiência do algoritmo MBM frente aos demais algoritmos.

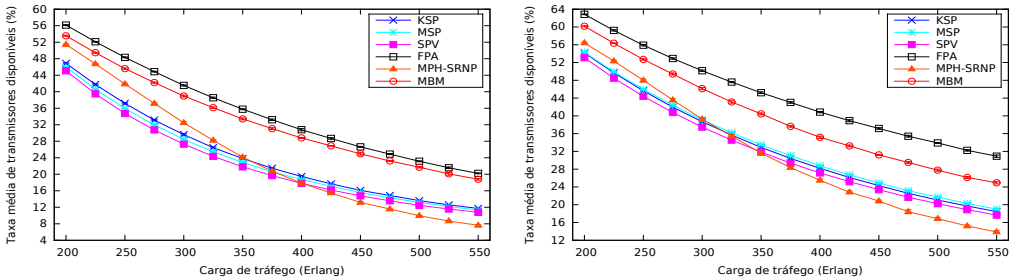


Figura 7. Taxa média de transmissores disponíveis. Topologias USANet e PanEuro, respectivamente

Média de saltos na topologia virtual por requisição

O número de saltos na topologia virtual indica o número de conversões OEO e processamento elétrico que são utilizados na rede. A Figura 8 mostra a média de saltos na topologia virtual por requisição para várias cargas de tráfego nas topologias UsaNet e PanEuro, respectivamente. Os algoritmos KSP, MSP, SPV e FPA mantêm em todas as cargas de tráfego uma média igual a 1, pois esses algoritmos são *singlehop*. Pode-se observar que o algoritmo MBM mantém sua média de saltos entre 2 e $2,5$ por requisição mesmo com o crescimento do tráfego. O mesmo não acontece ao algoritmo MPH-SRNP, onde

sua tendencia é aumentar o número de saltos com o crescimento do tráfego. Dessa forma, o algoritmo MBM faz, em média, dois saltos para atender uma requisição usando, em geral, o melhor nível de modulação possível. Observa-se que essa característica do MBM não afeta negativamente o desempenho do algoritmo, como demonstrado anteriormente.

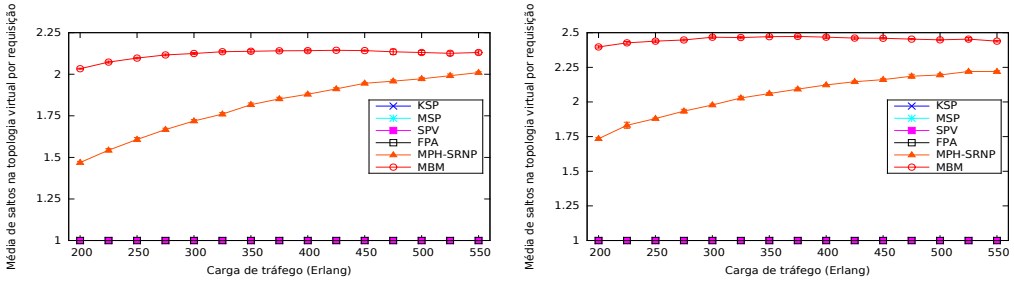


Figura 8. Média de saltos na topologia virtual por requisição. Topologias USANet e PanEuro, respectivamente

Taxa média de espectro disponível

A taxa de espectro disponível reflete o uso de recursos de espectro utilizado em toda a rede. A Figura 9 mostra a média da taxa de espectro disponível por requisição para várias cargas de tráfego nas topologias UsaNet e PanEuro, respectivamente. Observa-se que o algoritmo MBM tem o menor uso do espectro, em média cerca de 6% menor chegando a 8%, em todas as cargas de tráfego dentre todos os algoritmos avaliados, evidenciando o melhor desempenho do algoritmo MBM. Por aproveitar melhor o uso do espectro, o algoritmo MBM obtém uma melhor taxa de bloqueio que os demais. Um detalhe importante é que a mesma correspondência apresentada pela taxa de bloqueio de banda entre o algoritmo MBM e o FPA é refletida na taxa de uso do espectro, mostrando a importância desta métrica no bloqueio de banda.

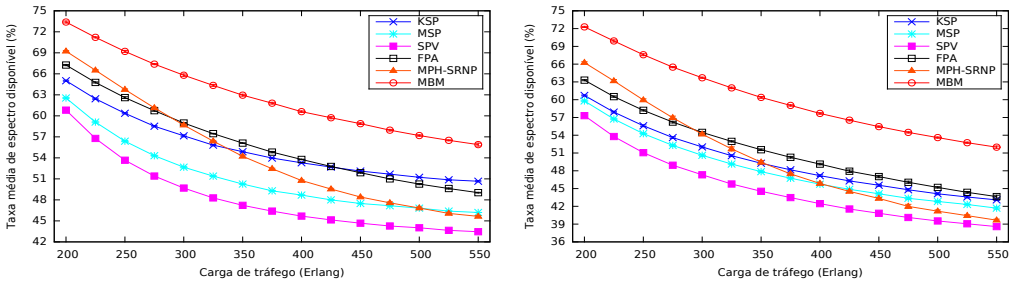


Figura 9. Taxa média de espectro disponível. Topologias USANet e PanEuro, respectivamente

Taxa média do uso de modulação

A taxa média do uso de modulação mostra o percentual de caminhos ópticos alocados para cada nível de modulação. A Tabela 2 mostra a taxa média do uso de modulação considerando a média para todas as cargas de trabalho nas topologias UsaNet e PanEuro, respectivamente. É necessário destacar que a taxa apresentou uma pequena variação entre as cargas de trabalho, com o desvios médios menores que 1. Os resultados mostram que o algoritmo MBM aproveita ao máximo o uso do maior nível de modulação fornecido

pelos transmissores. Com essa estratégia o algoritmo consegue um melhor desempenho no uso dos recursos de espectro proporcionando uma menor taxa de bloqueio de banda. A modulação 16QAM foi utilizada em 55% dos caminhos ópticos alocados na topologia UsaNet, e em 81% na topologia PanEuro. O algoritmo MPH-SRNP obteve um percentual bastante distribuído na topologia UsaNet. Na topologia PanEuro, o percentual maior concentrou-se na modulação 16QAM, uma vez que essa topologia possui uma distância média menor entre seus nós. Os demais algoritmos mantém o mesmo padrão de uso do nível de modulação em suas respectivas topologias concentrando seu maior percentual nas modulações QPSK e 8QAM para as topologias UsaNet e PanEuro, respectivamente, uma vez que, em média, modulações superiores não atendem as distâncias dos caminhos ópticos maiores.

Tabela 2. Taxa média do uso de modulação nas topologias USANet e PanEuro, respectivamente.

Algorithm	% BPSK	% QPSK	% 8QAM	% 16QAM
KSP	19,06	42,26	24,72	13,96
MSP	22,23	39,93	24,18	13,66
SPV	24,25	39,09	23,38	13,28
FPA	16,48	43,87	25,52	14,13
MPH-SRNP	27,87	26,39	20,49	25,25
MBM	7,95	18,42	18,57	55,06

Algorithm	% BPSK	% QPSK	% 8QAM	% 16QAM
KSP	1,37	37,75	39,78	21,10
MSP	2,77	36,64	39,23	21,36
SPV	3,82	38,14	37,48	20,56
FPA	1,19	37,24	40,36	21,21
MPH-SRNP	19,03	19,00	22,56	39,41
MBM	0,87	8,78	8,83	81,52

6. Conclusão

Este trabalho abordou as principais características do paradigma de redes ópticas elásticas (EON) e apresentou os principais desafios encontrados na literatura. Foi proposta uma técnica para solucionar o problema de roteamento e atribuição de espectro com modulação adaptativa (RMLSA) através do uso da agregação elétrica e óptica associado ao controle da modulação espectral no cenário de tráfego dinâmico. O objetivo de associar essas técnicas procura aproveitar melhor os recursos de espectro da rede óptica atendendo maiores taxas de dados. Para este fim, propõe-se o algoritmo MBM que procura realizar agregação óptica usando os níveis de modulação mais altos através de múltiplos saltos na topologia virtual. Dessa forma, para que uma transmissão óptica entre dois nós distantes seja realizada com o uso de um nível de modulação alto é necessário que essa transmissão óptica seja composta por vários caminhos ópticos mais curtos que possam atender os requisitos de QoT dos níveis de modulações mais elevados. Dessa maneira, é possível utilizar menos recursos de espectro ao custo de utilizar mais transmissores na rede.

Para demonstrar este fato, foram comparados cinco algoritmos da literatura com o algoritmo MBM em duas topologias de rede. Os resultados mostraram que o algoritmo MBM reduziu significativamente a taxa de bloqueio de banda (BBR) frente aos demais algoritmos da literatura. Embora o uso de transmissores tenha crescido cerca de 2% frente aos algoritmos comparados, observa-se que o MBM utiliza de maneira mais eficiente os transmissores através do roteamento *multihop*, viabilizando novos pontos de entrada na rede para a agregação elétrica e óptica, o que leva a um melhor uso de espectro na rede.

Referências

Christodoulopoulos, K., Tomkos, I., and Varvarigos, E. (2011). Elastic bandwidth allocation in flexible ofdm-based optical networks. *Journal of Lightwave Technology*, 29(9):1354–1366.

Drummond, A. C. (2016). Wdmsim: Wdm optical network simulator. http://www.lrc.ic.unicamp.br/wdmsim/.

Jinno, M., Kozicki, B., Takara, H., Watanabe, A., Sone, Y., Tanaka, T., and Hirano, A. (2010). Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [topics in optical communications]. *IEEE Communications Magazine*, 48(8):138–145.

Khodashenas, P. S., Comellas, J., Spadaro, S., and Perelló, J. (2013). Dynamic source aggregation of sub-wavelength connections in elastic optical networks. *Photonic Network Communications*, 26:2–3.

Saradhi, C. and Subramaniam, S. (2009). Physical layer impairment aware routing (pliar) in wdm optical networks: issues and challenges. *IEEE Communications Surveys Tutorials*, 11(4):109–130.

Tomkos, I., Azodolmolky, S., Sole-Pareta, J., Careglio, D., and Palkopoulou, E. (2014). A tutorial on the flexible optical networking paradigm: State of the art, trends, and research challenges. *Proceedings of the IEEE*, 102(9):1317–1337.

Wan, X., Hua, N., and Zheng, X. (2012). Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 4(8):603–613.

Wan, X., Wang, L., Hua, N., Zhang, H., and Zheng, X. (2011). Dynamic routing and spectrum assignment in flexible optical path networks. In *Optical Fiber Communication Conference and Exposition (OFC) and the National Fiber Optic Engineers Conference (NFOEC)*, pages 1–3.

Xiong, F. (2006). *Digital Modulation Techniques, Second Edition (Artech House Telecommunications Library)*. Artech House, Inc., Norwood, MA, USA.

Ye, Z., Patel, A., Ji, P., and Qiao, C. (2014). Distance-adaptive and fragmentation-aware optical traffic grooming in flexible grid optical networks. In *2014 OptoElectronics and Communication Conference and Australian Conference on Optical Fibre Technology*, pages 355–356.

Zhang, G., Leenheer, M. D., Morea, A., and Mukherjee, B. (2013). A survey on ofdm-based elastic core optical networking. *IEEE Communications Surveys Tutorials*, 15(1):65–87.

Zhang, G., Leenheer, M. D., and Mukherjee, B. (2012). Optical traffic grooming in ofdm-based elastic optical networks [invited]. *IEEE/OSA Journal of Optical Communications and Networking*, 4(11):B17–B25.

Zhang, J., Ji, Y., Song, M., Zhao, Y., Yu, X., Zhang, J., and Mukherjee, B. (2015). Dynamic traffic grooming in sliceable bandwidth-variable transponder enabled elastic optical networks. *Journal of Lightwave Technology*, PP(99):1–1.

Zhang, Y., Zheng, X., Li, Q., Hua, N., Li, Y., and Zhang, H. (2011). Traffic grooming in spectrum-elastic optical path networks. In *Optical Fiber Communication Conference and Exposition (OFC) and National Fiber Optic Engineers Conference (NFOEC)*, pages 1–3.

Trilha Principal do SBRC 2016
Sessão Técnica 19
Controle de acesso ao meio

Um protocolo FSA com resolução imediata de colisões para sistemas RFID sob o efeito captura

Henrique Stagni[1], **Rafael Perazzo Barbosa Mota**[1], **Daniel Macêdo Batista**[1]

[1]Departamento de Ciência da Computação – Universidade de São Paulo (USP)
Rua do Matão 1010 – 05508-090 – São Paulo – SP – Brasil

{stagni,perazzo,batista}@ime.usp.br

Abstract. *Capture effect is a common phenomenon in passive radio frequency identification systems. When multiple tags transmit their signals to a reader simultaneously, one of the tags will be successfully identified due to capture effect. In this paper, we present an algorithm which handles collisions as soon as they occur, which we believe to be a better way of taking advantage of a capture effect environment in order to improve identification efficiency. The algorithm we propose is both easy to implement and computationally efficient. Moreover we use a more realistic approach for modelling the capture effect. Our solution is based on the IDFSA-IcolR algorithm, a Frame Slotted Aloha (FSA) algorithm, with similar properties, which does not regard environments with capture effect.*

Resumo. *O efeito captura é um fenômeno comum em sistemas RFID. Quando múltiplas etiquetas respondem a uma solicitação simultaneamente, uma delas pode ser corretamente identificada devido a este efeito. Neste artigo é proposto um algoritmo que resolve colisões assim que elas ocorrem, o que é uma boa maneira de explorar um ambiente com efeito captura de forma a obter uma melhor eficiência de identificação. Ademais, o algoritmo proposto é de fácil implementação, é computacionalmente eficiente e modela o efeito captura de modo realístico. A proposta se baseia no algoritmo IDFSA-IcolR, um algoritmo FSA com propriedades similares mas que não considera efeito captura. Experimentos de simulação são realizados para comprovar as vantagens do algoritmo.*

1. Introdução

Identificação por radiofrequência (RFID) é um termo genérico para tecnologias que permitem a identificação de objetos por meio de ondas de rádio. Um número que identifica univocamente (ID) cada objeto de um sistema RFID é armazenado em uma *etiqueta* RFID. Uma etiqueta RFID é um dispositivo eletrônico que contém um microchip e uma antena. Tal dispositivo pode receber sinais de rádio de um *leitor* RFID e mandar, por sua vez, um sinal de volta ao leitor — usando uma técnica de reflexão de sinal conhecida como *retrodifusão* — contendo o seu número de identificação e, possivelmente, outras informações. Um *leitor* RFID é, também, um dispositivo eletrônico que, após ter identificado todas as etiquetas do sistema, envia tais informações para um outro dispositivo que se encarrega de efetivamente processá-las.

A tecnologia RFID possui muitas vantagens em relação a identificação pelos assim chamados *códigos de barra*, que ainda constituem, em diversas aplicações, a solução mais empregada para resolver o problema de identificação de objetos. Ao contrário de etiquetas

RFID, códigos de barra são pouco robustos — sujeira, umidade e dobras nas embalagens de produtos podem impedir sua leitura —, podem ser facilmente falsificados e, acima de tudo, exigem intervenção humana no processo de leitura.

A tecnologia RFID, por sua vez, não exige intervenção humana, possibilitando a identificação de múltiplas etiquetas de forma *eficiente*, isto é, em um curto intervalo de tempo. Ainda assim, existem aplicações que exigem uma eficiência de identificação que ainda não é atingida pelos atuais sistemas RFID, o que inviabiliza seu uso. A (relativamente) baixa eficiência dos sistemas RFID atuais é causada pela *colisão* entre etiquetas. Uma *colisão* ocorre quando múltiplas etiquetas respondem simultaneamente ao leitor. Como todas elas compartilham o mesmo canal de comunicação, os seus respectivos sinais de rádio se interferem mutuamente, o que geralmente impede a identificação de qualquer uma delas [1]. É possível, entretanto, que o leitor identifique uma das múltiplas etiquetas em colisão, como consequência de um fenômeno chamado *efeito captura*. O *efeito captura* ocorre quando o sinal proveniente de alguma das etiquetas em colisão tem uma potência muito maior do que o sinal das demais de modo que o leitor só consegue receber o sinal da etiqueta que *captura* o canal, isto é, aquela com a maior potência. Apesar de ser um fenômeno muito observado na prática, o *efeito captura* em sistemas RFID só tem sido considerado em trabalhos mais recentes (ver Seção 2).

Um *protocolo anticolisão* para um sistema RFID é uma especificação do conjunto de *mensagens* (ou *comandos*) que o leitor pode enviar às etiquetas, bem como a ação tomada por uma *etiqueta* ao receber cada uma dessas mensagens — geralmente o envio de uma mensagem para o leitor e uma mudança de *estado*. O *protocolo* também especifica o tamanho (em número de *bits*) dos identificadores de cada etiqueta, o código de detecção de erros usado para constatar a ocorrência de *colisões*, e diversos outros parâmetros específicos do sistema em questão. Finalmente temos o algoritmo anticolisão, responsável por ditar a estratégia usada pelo leitor para enviar mensagens específicas de um protocolo anticolisão em questão, de forma a reconhecer todas as etiquetas do sistema no menor tempo possível, resolvendo as prováveis colisões que venham a ocorrer.

Neste artigo propomos o IDFSA-IcolR-CE, uma solução para identificação de etiquetas RFID em ambientes sob o efeito captura. Nossa solução compreende um algoritmo anticolisão FSA (*Frame Slotted Aloha* — ver seção 2) com *resolução imediata* de colisões, isto é, em que etiquetas que participam de uma colisão são reconhecidas em um processo que não envolve as demais etiquetas e que é iniciado imediatamente após a colisão. Nosso algoritmo foi concebido para um sistema RFID abstrato, dotado de um protocolo que possibilita que um leitor identifique a ocorrência de efeito captura assim que ele ocorre.

Na seção seguinte, revisamos os algoritmos anticolisão FSA existentes na literatura. Na Seção 3 descrevemos o modelo que assumimos para o efeito captura e outras suposições que fazemos neste trabalho. A Seção 4 descreve o protocolo desenvolvido e a Seção 5 descreve o algoritmo anticolisão específico usado por nossa solução. Os resultados obtidos são apresentados na Seção 6 e discutidos na Seção 7.

2. Algoritmos anticolisão FSA

Neste artigo serão considerados sistemas RFID pertencentes à classe FSA (*Frame Slotted Aloha*). Tais sistemas possuem protocolos que permitem que o leitor envie uma men-

sagem que determina o início de um novo *quadro*, composto de um dado número L de etapas distintas de comunicação denominadas *slots*— o início de cada um desses *slots* é determinado por uma mensagem enviada pelo leitor. Fixado um quadro, cada etiqueta pode se comunicar com o leitor em apenas um *slot*, escolhido uniformemente ao acaso pela etiqueta no início do quadro. Para cada *slot*, definimos seu *tamanho de colisão* como o número de etiquetas associados a ele. Dizemos que um *slot* com tamanho de colisão s é *ocioso*, de *identificação* ou de *colisão* se $s = 0$, $s = 1$ ou $s > 1$, respectivamente. Diferentes sistemas FSA se distinguem principalmente pelo algoritmo anticolisão empregado, mas também por particularidades dos protocolos anticolisão.

Podemos classificar os algoritmos FSA de acordo com o momento em que as etiquetas colididas são identificadas. Em algoritmos *DFSA* (*Dynamic Frame Slotted Aloha*), o leitor repete sucessivos quadros (possivelmente com diferentes tamanhos) até que todas as etiquetas tenham sido identificadas. Um exemplo peculiar de algoritmo DFSA é o chamado Algoritmo Q para o padrão global EPC (*EPC Class 1 Generation 2* [2]). O protocolo especificado no padrão EPC permite que o leitor mande uma mensagem que inicia um novo quadro antes do término do atual. O Algoritmo Q (o algoritmo padrão especificado no EPC) usa tais mensagens para iniciar um novo quadro (de tamanho possivelmente diferente do anterior) a cada *slot* de colisão. Uma outra classe de algoritmo FSA são os algoritmos *com resolução imediata de colisão*. Tais algoritmos mantêm um único *quadro principal* e identificam etiquetas colididas em um mecanismo separado enquanto as demais etiquetas esperam. O algoritmo proposto em [3], por exemplo, reconhece etiquetas colididas em um mecanismo conhecido como *árvore binária*. A solução proposta neste trabalho é baseada no algoritmo IDFSA-IcolR [4], no qual todo *slot* de colisão do quadro principal dá origem à criação de um *quadro de colisão*, em que participam apenas as etiquetas colididas. Durante a execução de um quadro de colisão, o algoritmo se comporta como um sistema DFSA, repetindo o quadro (com um tamanho possivelmente diferente) até que todas as etiquetas sejam identificadas e o algoritmo possa voltar à execução do quadro principal.

Um ponto crítico de todo algoritmo FSA é a escolha do tamanho dos quadros usados. Um tamanho de quadro muito menor que o número de etiquetas tende a conter muitos *slots* de colisão, enquanto que um quadro de tamanho muito maior tende a conter muitos *slots* ociosos. Assim, todos os algoritmos FSA devem fazer alguma espécie de estimativa do número de etiquetas do sistema. Em alguns algoritmos, como o proposto neste trabalho, a estimativa precede o processo de identificação. Em outros, como o Algoritmo Q, a estimativa é feita ao longo do próprio processo de identificação, uma vez que o tamanho do quadro pode ser alterado a qualquer momento.

Apesar da grande variedade de soluções para sistemas RFID propostas na literatura, há poucos trabalhos que lidam com o fenômeno do efeito captura. Algoritmos DFSA em geral já são capazes de identificar todas as etiquetas do sistema, mesmo considerando o efeito captura. Entretanto, a ocorrência desse fenômeno pode fazer com que tais algoritmos façam uma estimativa errada do número de etiquetas, degradando a eficiência da solução [5]. Li e Wang propõe em [6] um algoritmo de estimação que lida com efeito captura, permitindo que algoritmos DFSA que façam uso dessa estimação não subestimem o número de etiquetas do sistema. Wu e Zeng [5] desenvolveram um algoritmo anticolisão compatível com o padrão EPC que lida com o efeito captura. Embora esse sistema identi-

fique etiquetas que colidiram em *slots* sem efeito captura imediatamente, as etiquetas em *slots* de colisão capturados são identificadas apenas em quadros subsequentes.

Este trabalho se distingue dos demais ao propor um sistema RFID em que *todas* as colisões são resolvidas imediatamente, independente da eventual ocorrência de efeito captura. Ademais, a exemplo dos dois sistemas citados anteriormente, boa parte dos trabalhos existentes assume que o efeito captura pode ocorrer em cada *slot* com uma probabilidade fixa. Nossa solução assume um modelo mais realístico para o efeito captura, que será descrito na próxima seção.

3. Suposições sobre o sistema

3.1. Modelo de efeito captura

Neste trabalho, usamos um modelo mais realístico [7] para o efeito captura, em vez do modelo mais simples usado por boa parte dos algoritmos atualmente (ver Seção 2).

Em experimentos práticos, o efeito captura costuma ocorrer com maior frequência quando um *slot* tem um *tamanho de colisão* pequeno, isto é, quando apenas um número pequeno de etiquetas é associado a ele [8, 9]. Esse fenômeno pode ser inclusive verificado analiticamente se assumirmos que as etiquetas foram dispostas, uniformemente ao acaso, no interior de uma região circular [9, 10]. Os dois artigos supracitados, apesar de assumirem respectivamente diferentes modelos estocásticos para a propagação por rádio, provam que a probabilidade do efeito captura ocorrer em um dado *slot* de tempo decresce *exponencialmente* com o tamanho de colisão do *slot*. Em outras palavras, podemos assumir que há uma constante positiva γ tal que a probabilidade $C(s)$ de um *slot* com tamanho de colisão s ser capturado é dada pela equação (1) abaixo.

$$C(s) = e^{-\gamma(s-1)}. \tag{1}$$

Note que a equação acima satisfaz $C(1) = 1$, de forma que podemos considerar um *slot* de identificação usual (com tamanho de colisão 1) como uma forma degenerada de efeito captura.

3.2. Suposições gerais

O sistema consiste de um leitor RFID e n etiquetas passivas satisfazendo as condições listadas abaixo.

- As n etiquetas estão na área de cobertura do leitor e não se movem durante o processo de identificação. Nenhuma nova etiqueta entra na área de cobertura do leitor enquanto o processo de identificação estiver em andamento.
- Não há perdas ou atrasos na transmissão de quaisquer mensagens do sistema.
- O parâmetro γ da Equação 1 é conhecido pelo sistema. Fazemos tal suposição, uma vez que tal parâmetro não varia quando o processo de identificação é repetido em um mesmo local e com equipamentos similares.
- Supomos que o leitor *não* possui meios de distinguir entre um *slot* de identificação e um *slot* de colisão em que ocorre efeito captura.

4. Protocolo anticolisão

Nesta seção, especificamos um protocolo anticolisão cujo objetivo é dar suporte a algoritmos anticolisão com resolução imediata de colisões. Em particular, nosso protocolo fornece meios para que um algoritmo anticolisão reconheça se o *slot* corrente é de colisão, de forma que o processo de reconhecimento das etiquetas em colisão possa ser iniciado imediatamente.

O protocolo proposto assume que cada mensagem das etiquetas para o leitor contém um *código de detecção de erros* (como CRC), de forma que o leitor consiga identificar quando há uma interferência de sinal. Contudo, só isso não é suficiente para que o leitor distingua um *slot* de colisão em que ocorre efeito captura de um *slot* de identificação. A solução do protocolo proposto consiste em uma troca extra de mensagens entre as etiquetas e o leitor em *slots* em que uma etiqueta é reconhecida: após informar o reconhecimento da etiqueta, o leitor espera possíveis mensagens de outras etiquetas informando que estavam associadas a esse mesmo *slot* (o que configura a ocorrência do efeito captura). Observamos que se nessa transmissão extra ocorre novamente efeito captura (possivelmente degenerado), o leitor pode ser capaz de reconhecer uma etiqueta adicional. Chamaremos tal identificação de *identificação tardia*.

O protocolo proposto dá suporte a dois tipos de quadro: um quadro *principal* — do qual participam inicialmente todas as etiquetas — e quadros *de colisão* — dos quais participam apenas as etiquetas associadas ao último *slot* processado no quadro principal. O protocolo permite ainda que um quadro de colisão seja repetido diversas vezes, recurso que pode ser usado por algoritmos anticolisão até que todas as etiquetas colididas sejam identificadas.

4.1. Mensagens do protocolo

O leitor pode trocar as mensagens da Tabela 1 com as etiquetas. Cada etiqueta pode enviar as mensagens listadas na Tabela 2 como resposta a uma comunicação vinda do leitor.

4.2. Diagrama de estados da etiqueta

O algoritmo da etiqueta é dado pelo diagrama de estados da Figura 1. Na figura, `ID` representa o identificador da etiqueta, o símbolo $\star$ representa qualquer identificador diferente de `ID` e `unif(1..L)` é a operação de escolher um número uniformemente ao acaso entre 1 e L. `rec:` representa uma mensagem recebida e `env:` representa uma mensagem enviada.

4.3. Tamanho ótimo do quadro principal

Nesta seção analisamos a *eficiência* do quadro principal, dado pelo número médio de etiquetas identificadas por *slot* desse quadro. Essa análise determina o tamanho de quadro principal que um algoritmo anticolisão qualquer pode usar para maximizar o número de identificações no quadro principal. Lembramos que é possível que até duas etiquetas sejam reconhecidas em um mesmo *slot*, como consequência de uma eventual identificação tardia.

Suponha que o sistema tenha n etiquetas e que o tamanho do quadro principal seja L. Fixe um *slot* i qualquer do quadro principal e defina S como a variável aleatória que

Nome da Mensagem	O que indica
QUADRO(L)	Início da fase de identificação. O parâmetro L é um inteiro positivo que indica o tamanho do quadro principal.
SLOT(i)	Início do *slot* i.
ACK(i, ID)	Leitor reconheceu etiqueta com identificador ID durante *slot* i do quadro principal.
QUADROCOL(i, L, $[ID]$)	Início de um quadro de colisão de tamanho L para reconhecer as etiquetas associadas ao *slot* i que não foram identificadas no quadro principal. O parâmetro opcional ID indica que uma etiqueta com identificador ID foi reconhecida tardiamente no *slot* i do quadro principal.
SLOTCOL(i, $[ID]$)	Início do *slot* i do quadro de colisão. O parâmetro opcional ID indica que uma etiqueta com identificador ID foi reconhecida tardiamente no *slot* de colisão $i - 1$.
ACKCOL(i, ID)	Leitor reconheceu etiqueta com identificador ID durante *slot* i do quadro de colisão.

Tabela 1. Mensagens entre leitor e etiquetas.

Nome da Mensagem	O que indica
ETIQUETA(ID)	Informa ao leitor o número ID que a identifica.
CAPTURADA(ID)	Informa ao leitor que esta etiqueta estava associada ao mesmo *slot* que a última etiqueta identificada.

Tabela 2. Mensagens de resposta das etiquetas.

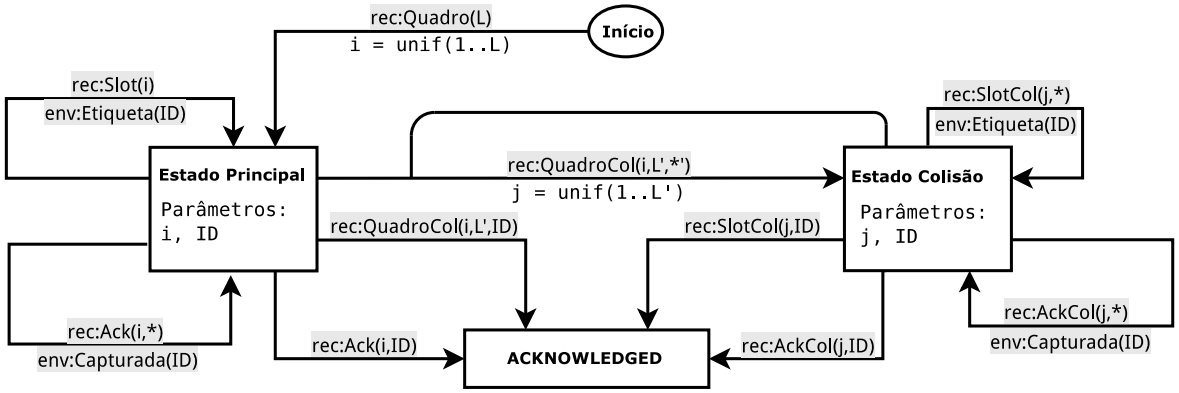


Figura 1. Diagrama de estados de uma etiqueta com identificador `ID`

conta o número de etiquetas associadas a i. Considere também a variável aleatória Z que indica o número de etiquetas *reconhecidas* durante o *slot* i.

A eficiência média ef do sistema (no quadro principal) pode ser expressa, portanto, como

$$\begin{aligned} ef &= \mathbb{E}(Z) \\ &= 1 \cdot \mathbb{P}(Z=1) + 2 \cdot \mathbb{P}(Z=2) \\ &= 1 \cdot \Big(\mathbb{P}(S=1) + \sum_{s=2}^{n} \mathbb{P}(S=s) C(s)(1 - C(s-1)) \Big) \\ &\quad + 2 \cdot \Big(\sum_{s=2}^{n} \mathbb{P}(S=s) C(s) C(s-1) \Big) \\ &= \mathbb{P}(S=1) + \sum_{s=2}^{n} \mathbb{P}(S=s) C(s)(1 + C(s-1)) \end{aligned}$$

Como cada etiqueta é associada ao *slot* i com probabilidade $\frac{1}{L}$ e de maneira independente da escolha das demais etiquetas, temos que S tem distribuição binomial, isto é, que $S \sim Bin(n, \frac{1}{L})$. Portanto, segue que

$$ef = \frac{n}{L} \left(1 - \frac{1}{L}\right)^{n-1} + \sum_{s=2}^{n} \binom{n}{s} \left(\frac{1}{L}\right)^{s} \left(1 - \frac{1}{L}\right)^{n-s} e^{-\gamma(s-1)}(1 + e^{-\gamma(s-2)}) \quad (2)$$

Fixado γ, desconhecemos uma fórmula fechada que determine o valor de L que maximize a expressão acima em função de n. Notamos, entretanto, que esse valor pode ser computado *a priori* para cada n por força bruta e armazenado em uma tabela. Desta forma, um algoritmo anticolisão que tenha acesso a tal tabela pode escolher o melhor tamanho de quadro principal em função de uma estimativa para o número n de etiquetas. No caso particular do protocolo IDFSA-IcolR-CE, os valores estimados para o número de etiquetas são sempre potências de 2, o que permite que o tamanho dessa tabela seja drasticamente reduzido.

5. Algoritmo anticolisão IDFSA-IcolR-CE

O algoritmo anticolisão proposto possui duas fases distintas. A primeira consiste na estimação do número de etiquetas do sistema, de forma a determinar um bom valor para o quadro principal, e a segunda consiste na identificação propriamente dita.

5.1. Fase de estimação

A fase de estimação considerada aqui é inspirada no processo de estimação usado em [4] e tem o objetivo de determinar rapidamente apenas uma estimativa grossa do número de etiquetas do sistema. Mais especificamente, esse algoritmo determina uma estimativa Q para o maior inteiro Q^* que satisfaça $2^{Q^*} \leq n$, onde n é o número de etiquetas do sistema.

O algoritmo inicia tomando $Q = 1$. A cada passo, o algoritmo de estimação repete um número T de vezes o seguinte experimento: cria um quadro de tamanho $L = 2^Q$ e chama apenas as etiquetas associadas ao primeiro *slot*. Se houve colisão (ou efeito

captura) em mais do que 40% dessas T vezes, o algoritmo de estimação dobra o valor de Q. Caso contrário, o algoritmo devolve o valor atual de Q como sua estimativa para Q^*.

De fato, se em algum passo do algoritmo temos $Q < Q^*$, então $L < 2n$, o que implica que a probabilidade de que mais do que uma etiqueta esteja associada ao primeiro *slot* é pelo menos $1 - 3e^{-2} \approx 59\%$, quando $n \to \infty$. Por outro lado, se $Q \geq Q^*$, então $L \geq 2n$ e, portanto, a probabilidade de que mais do que uma etiqueta esteja associada ao primeiro *slot* é no máximo $1 - 2e^{-1} \approx 26\%$, quando $n \to \infty$.

5.2. Fase de identificação

Nessa seção apresentamos na forma de pseudocódigo (Algoritmo 1 e Algoritmo 2) o algoritmo usado pelo leitor na fase de identificação. Assumimos que o leitor tem disponível uma tabela com os valores ótimos de quadro principal para cada n. Essa tabela pode ser computada previamente como descrito na Seção 4.3. O código que implementa os algoritmos está disponível em `https://github.com/stagni/icolrce` sob a licença GPL v2.0.

Algoritmo 1 Operação principal do leitor

```
 1: n ← (valor obtido a partir do processo de estimação).
 2: L ← TamanhoQuadroOtimo(n)
 3: Envia mensagem QUADRO(L).
 4: para i = 1 to L faça
 5:     Envia mensagem SLOT(i).
 6:     se (não recebe respostas) então
 7:         Nada a fazer (slot ocioso)
 8:     senão se (recebe sinal com colisão) então
 9:         Chama ResolveColisões(i, 2, nil) (Algoritmo 2)
10:     senão se (recebe mensagem ETIQUETA(ID)) então
11:         Envia mensagem ACK(i, ID)
12:         se (não recebe respostas) então
13:             Nada a fazer (não ocorreu efeito captura)
14:         senão se (recebe mensagem CAPTURA(ID')) então
15:             Chama ResolveColisões(i, 1, nil) (Algoritmo 2)
16:         senão se (recebe sinal com colisão) então
17:             Chama ResolveColisões(i, 1, nil) (Algoritmo 2)
18:         fim se
19:     fim se
20: fim para
```

Fazemos aqui algumas observações sobre o tamanho do quadro de colisão usado pela nossa solução. Esse quadro possui apenas 2 *slots* se foi originado de uma colisão normal e apenas 1, se foi originado de uma colisão com efeito captura. A escolha se justifica pelo fato do número médio de etiquetas associadas a um mesmo *slot* do quadro principal ser pequeno. Além disso, na ocorrência de efeito captura pelo menos uma das etiquetas colididas já foi identificada. Observamos na prática que em um número considerável de vezes (principalmente na ocorrência de identificações tardias), não resta nenhuma etiqueta a ser identificada no quadro de colisão.

O tamanho do quadro de colisão é aumentado por um fator multiplicativo de $1{,}1$ no caso em que nem todas as etiquetas foram identificadas e o processo precisa ser repetido. Esse aumento é feito apenas para garantir que a eficiência do sistema não seja deteriorada no caso raro em que em que muitas etiquetas participam do quadro de colisão: neste caso, o número de *slots* no quadro de colisão eventualmente cresce até um valor que permita a identificação de tais etiquetas.

Algoritmo 2 Função de tratamento de *slot* de colisão.

```
1:  procedure RESOLVECOLISÕES(i, L, idTardia)
2:      Envia mensagem QUADROCOL(j, ⌊L⌋, idTardia).
3:      idTardia ← nil.
4:      repita
5:          colisão ← false
6:          para j = 1 até ⌊L⌋ faça
7:              Envia mensagem SLOTCOL(j, idTardia).
8:              idTardia ← nil.
9:              se (não recebe respostas) então
10:                 Nada a fazer (slot ocioso)
11:             senão se (recebe sinal com colisão) então
12:                 colisão ← true
13:             senão se (recebe mensagem ETIQUETA(ID)) então
14:                 Envia mensagem ACKCOL(j, ID)
15:                 se (não recebe respostas) então
16:                     Nada a fazer (não ocorreu efeito captura)
17:                 senão se (recebe mensagem CAPTURA(ID′)) então
18:                     colisão ← true.
19:                     idTardia ← ID′
20:                 senão se (recebe sinal com colisão) então
21:                     colisão ← true.
22:                 fim se
23:             fim se
24:         fim para
25:         se colisão = true então
26:             L ← 1.1 × L
27:             Envia mensagem QUADROCOL(i, ⌊L⌋, nil)
28:         fim se
29:     até colisão = false
30: fim procedure
```

6. Experimentos

Os experimentos realizados para avaliar o desempenho da solução proposta foram feitos por meio de simulações computacionais [11]. Para efeito de comparação, implementamos o Algoritmo Q, como especificado no padrão EPC [2], que é capaz de identificar todas as etiquetas, mesmo sob a ocorrência do efeito captura (como descrito na Seção 2). Como as demais propostas da literatura não utilizam um modelo realístico de efeito captura, os mesmos não foram considerados nas comparações para evitar comparações injustas. O

cenário modelado considera uma única identificação, com variação no número de etiquetas. O grande número de etiquetas foi considerado prevendo um aumento significativo da quantidade de objetos no futuro, principalmente em cenários da Internet das Coisas.

6.1. Métricas utilizadas

A principal métrica utilizada na literatura para avaliar o desempenho de sistemas RFID é o número de *slots* usados durante todo o processo de identificação. De maneira equivalente, também considera-se a *eficiência do sistema*, que é dada pelo número médio de identificações realizadas em cada *slot*.

Uma outra métrica utilizada diz respeito ao tempo total gasto durante o processo de identificação. Essa métrica não é equivalente à anterior, pois o tempo gasto em um *slot* pode variar se for um *slot* de identificação, de colisão ou ocioso. Neste trabalho consideramos que o leitor consome um tempo médio de $1,375ms$, $0,3375ms$ ou $0,0675ms$ em *slots* de identificação, de colisão ou ocioso, respectivamente. Apesar de serem relativos a leitores compatíveis com o padrão EPC (ver [12]), esses valores de tempo médio representam a relação de tempos válida em qualquer sistema RFID, a saber, de que *slots* de identificação são mais custosos que os demais tipos de *slot*.

É importante notar que em *slots* de identificação, há um comunicação extra realizada pelo IDFSA-IcolR-CE para que o leitor fique ciente da ocorrência do efeito captura. Assim, o tempo gasto pelo IDFSA-IcolR-CE em *slots* de identificação é potencialmente maior do que o tempo gasto por soluções que não necessitam desta comunicação extra (como o Algoritmo Q). As métricas descritas não são, portanto, as métricas ideais para comparar os algoritmos IDFSA-IcolR-CE e Algoritmo Q. Fazemos uma discussão mais aprofundada sobre essa questão na Seção 7.

6.2. Resultados obtidos

Na Figura 2 comparamos a eficiência (isto é, o número médio de identificações por *slot*) das soluções IDFSA-IcolR-CE e Algoritmo Q para um valor fixo de γ ($\gamma = 0{,}5$) e para diferentes números (N) de etiquetas a serem identificadas. Observamos que ao considerarmos um número razoavelmente grande de etiquetas, o sistema IDFSA-IcolR-CE obtém uma eficiência superior à obtida pelo Algoritmo Q. O intervalo de confiança estatístico foi de 95% e encontrou-se variações no intervalo de $0,3\%$ a $0,8\%$ do valor da média para todos os pontos. Para facilitar a leitura dos gráficos, não foram incluídos os intervalos de confiança, uma vez que os mesmos são pequenos o suficiente para serem seguramente ignorados. Assume-se que perdas, atrasos e outros erros são tratados em camadas mais baixas, de forma que o processo anticolisão se concentre no problema de minimizar o número de colisões durante o processo de identificação.

Também comparamos a eficiência dos algoritmos IDFSA-IcolR-CE e Q ao considerarmos diferentes valores de γ. Na Figura 3 comparamos o número de *slots* gastos por cada sistema no processo de identificação de 1000 etiquetas. O sistema proposto obtém um desempenho superior ao Algoritmo Q para todos os valores de γ considerados. Como esperado, a medida que γ diminui, o número de *slots* gastos no processo de identificação diminui em ambos os sistemas, pois estamos considerando ambientes onde a ocorrência de efeito captura é cada vez mais provável.

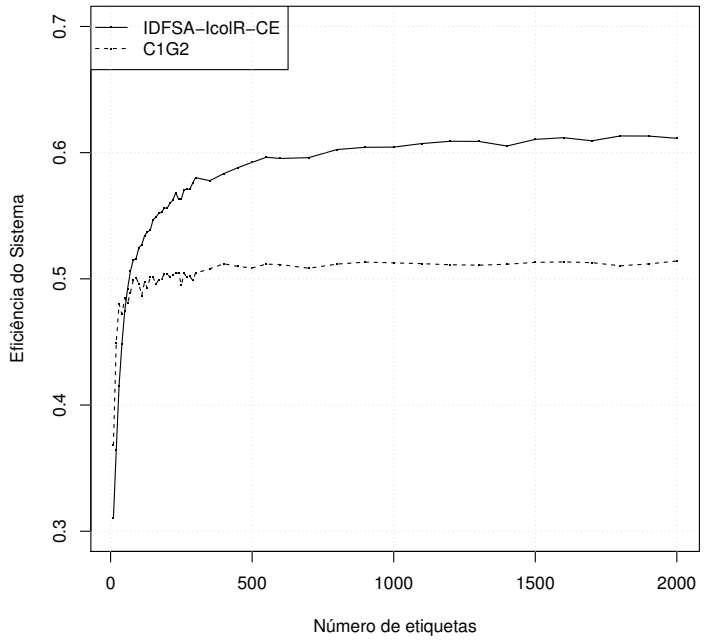


Figura 2. Número médio de identificações por *slot* ($\gamma = 0.5$)

Também obtemos um melhor desempenho usando a nossa solução quando comparamos o número de segundos gastos no processo de identificação. A Figura 4 mostra a redução relativa no tempo de identificação de 1000 etiquetas ao usarmos nossa solução em vez do Algoritmo Q.

7. Conclusão

Os resultados obtidos mostram que o algoritmo IDFSA-IcolR-CE obtém um desempenho consideravelmente melhor do que o Algoritmo Q ao identificar um número considerável de etiquetas em ambientes com efeito captura. Esta maior eficiência da nossa solução decorre, em parte, da escolha das métricas usadas: ao equipararmos o custo de *slots* de identificação em ambas as soluções, estamos desconsiderando a comunicação extra exigida pela nossa solução para que o leitor fique ciente da (possível) ocorrência de efeito captura. Por mais que o protocolo descrito no EPC também faça, na prática [2], a mesma quantidade de trocas de mensagens em um *slot* de identificação, esta troca extra tem como objetivo diminuir a quantidade de *bits* transmitidos entre leitor e etiquetas e o sistema IDFSA-IcolR-CE exigiria, de qualquer maneira, uma transmissão de uma maior quantidade de dados em um *slot* de identificação.

Como a resolução instantânea de colisões em cenários de efeito captura exige que *slots* de identificação sejam mais custosos, concluímos que é necessário o uso de métricas mais finas do que as geralmente encontradas na literatura para que esse tipo de solução seja avaliado.

Contudo, é importante ressaltar que o ganho de eficiência de nossa solução em relação ao Algoritmo Q é cada vez maior, à medida em que consideramos ambientes

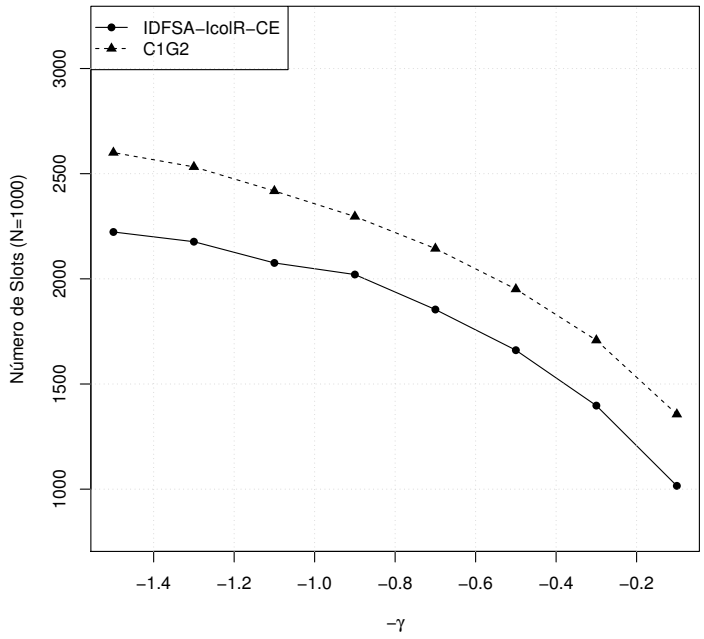


Figura 3. Número total de *slots* gastos no processo de identificação de 1000 etiquetas

com maior incidência de efeito captura. Isso indica que a maior eficiência do sistema IDFSA-IcolR-CE decorre não apenas das métricas que o favorecem, mas também da sua capacidade de explorar ambientes em que há maior ocorrência de efeito captura.

Finalmente, observamos que o valor ótimo obtido para o tamanho dos quadros de colisão (que, ao contrário do quadro principal, foram determinados empiricamente) foi extremamente pequeno (ver Algoritmo 2). Isso indica que uma maior eficiência poderia ser obtida ao empregarmos outros métodos para reconhecer etiquetas colididas, como por exemplo o uso de árvores binárias [3].

Referências

[1] D. Klair, K.-W. Chin, and R. Raad, "On the energy consumption of pure and slotted aloha based {RFID} anti-collision protocols," *Computer Communications*, vol. 32, no. 5, pp. 961 – 973, 2009.

[2] EPCglobal, GS1 Inc., "EPCTM Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 2.0.1," EPCGlobal Inc., Tech. Rep., April 2015, Acesso em 25 de Dezembro de 2015. [Online]. Available: \url{http://www.gs1.org/epcrfid/epc-rfid-uhf-air-interface-protocol/2-0-1}

[3] H. Wu, Y. Zeng, J. Feng, and Y. Gu, "Binary tree slotted aloha for passive rfid tag anticollision," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 19–31, 2013.

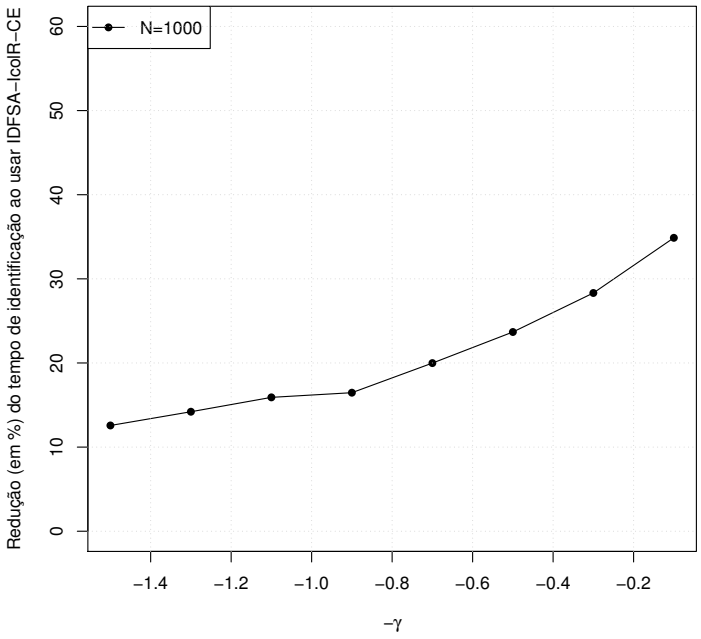


Figura 4. Redução relativa do tempo gasto na identificação de 1000 etiquetas: $(T_Q - T_{IcolR-CE})/T_Q$

[4] R. P. B. Mota, "An Easy to Implement and Energy-saving Anti-Collision Algorithm for RFID Systems," in *Proceedings of the Application of Information and Communication Technologies (AICT), 2015 IEEE 9th International Conference on*, ser. AICT '15. Rostov on Don, Rostov, Russia: IEEE, 2015.

[5] H. Wu and Y. Zeng, "Passive RFID Tag Anticollision Algorithm for Capture Effect," *Sensors Journal, IEEE*, vol. 15, no. 1, pp. 218–226, January 2015.

[6] B. Li and J. Wang, "Efficient Anti-Collision Algorithm Utilizing the Capture Effect for ISO 18000-6C RFID Protocol," *Communications Letters, IEEE*, vol. 15, no. 3, pp. 352–354, March 2011.

[7] A. Bekkali, S. Zou, A. Kadri, M. Crisp, and R. Penty, "Performance analysis of passive uhf rfid systems under cascaded fading channels and interference effects," *Wireless Communications, IEEE Transactions on*, vol. 14, no. 3, pp. 1421–1433, March 2015.

[8] M. Zorzi, "Capture probabilities in random-access mobile communications in the presence of rician fading," *Vehicular Technology, IEEE Transactions on*, vol. 46, no. 1, pp. 96–101, Feb 1997.

[9] J. Sanchez and D. Smith, "Capture effect in rician fading channels with application to slotted aloha," in *Global Telecommunications Conference, 1999. GLOBECOM '99*, vol. 5, 1999, pp. 2390–2394 vol.5.

[10] J. C. Arnbak and W. Van Blitterswijk, "Capacity of slotted aloha in rayleigh-fading channels," *Selected Areas in Communications, IEEE Journal on*, vol. 5, no. 2, pp. 261–269, 1987.

[11] R. P. B. Mota, "Simulator and Benchmark for RFID Anti-Collision Evaluation," in *Proceedings of the Application of Information and Communication Technologies (AICT), 2015 IEEE 9th International Conference on*, ser. AICT '15. Rostov on Don, Rostov, Russia: IEEE, 2015.

[12] W. Chen, "Optimal frame length analysis and an efficient anticollision algorithm with early adjustment of frame length for rfid systems," *Vehicular Technology, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

Um Protocolo de Acesso ao Meio com Network Coding em Ambiente PLC.

Luã M. F. Silveira[1],Roberto M. Oliveira[2], Moises V. Ribeiro[2]
Luiz Filipe M. Vieira[3], Marcos Augusto M. Vieira[3], Alex B. Vieira[1]

[1]Departamento de Ciência da Computação, Universidade Federal de Juiz de Fora
[2]Departamento de Engenharia Elétrica, Universidade Federal de Juiz de Fora
[3]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais

lmfsilveira@ice.ufjf.br, {rmassi.oliveira,mribeiro,}@engenharia.ufjf.br
{lfvieira, mmvieira}@dcc.ufmg.br, alex.borges@ufjf.edu.br

Abstract. *In this paper, we introduce the use of network coding, in the medium access control layer (MAC), in a power line network environment. We propose a MAC layer protocol and evaluate its performance thought simulation of common topology for a PLC system with time division multiple access (TDMA-OFDM). The protocol we propose uses the assistance of a relay, which is responsible for encoding the MPDU's. Our results show that, in a broadcast like transmission, the use of network coding can increase the overall goodput an average of 115% compared to a traditional stop&wait scenario. Moreover, our protocol reduces in 112% the average occupancy of network buffers. Finally, mean latency in a system using the protocol we propose is 4 times lower.*

Resumo. *Este artigo apresenta um caso de utilização de network coding, na camada de controle de acesso ao meio (MAC), em ambientes com comunicação via rede elétrica (PLC). Nós propomos e avaliamos o desempenho de um protocolo a partir de simulações de uma topologia comum em um sistema PLC com acesso múltiplo por divisão no tempo (TDMA-OFDM). O protocolo proposto utiliza o auxílio de um relay responsável de codificação dos MPDU's. Nossos resultados apontam que, em uma transmissão tipo broadcast, o uso de network coding pode aumentar o goodput médio em até 115% em relação a um cenário tradicional equivalente, com uso de protocolos estilo stop&wait. Mais ainda, o novo protocolo diminui a ocupação média dos buffers dos dispositivos de rede em até 112%. Finalmente, as latências medianas observadas no sistema com o protocolo proposto são até 4 vezes menores.*

1. Introdução

Comunicação via rede elétrica (PLC) atrai cada vez mais a atenção, tanto da comunidade acadêmica, quanto da indústria. A demanda por esse tipo de rede se dá por sua flexibilidade/facilidade na conexão de dispositivos, baixo custo quando comparado a tecnologias sem fio e, até mesmo, por suas altas taxas de transmissão [Ezzine et al. 2014]. Como consequência desse interesse, cada vez mais nos deparamos com redes residenciais e *backbones* de alta velocidade baseadas nesse tipo de rede.

Ainda hoje, técnicas para transmissão de dados utilizadas em PLC podem apresentar altas taxas de falha, dado uma forte atenuação do sinal de comunicação [Gao et al.

2008]. Por esse motivo, há uma grande discussão em torno da utilização de um protocolo de cooperação adequado para melhorar o desempenho da comunicação. A maior parte das contribuições nessa área discute os tipos de melhorias que podem ser obtidas na camada física, utilizando cooperação com técnicas de combinação de sinais provenientes de canais distintos [Biagi 2011, Cheng et al. 2013, Kim et al. 2012, Valencia et al. 2014]. No entanto, há a possibilidade de implementação, tanto de técnicas de cooperação, quanto de *network coding* (NC), em camadas superiores [Noori and Lampe 2013, Bilbao et al. 2014]. Apesar das diversas oportunidades e frentes de trabalho em PLC, ainda há a prevalência de protocolos convencionais, baseados em mestre-escravo ou inundação do canal de comunicação [Gao et al. 2008].

Nesse sentido, dada a natureza de falhas do cenário PLC e os notórios benefícios de *network coding* na melhoria do desempenho de redes, nós propomos um mecanismo de *network coding* na camada de acesso ao meio (MAC) em PLC. Nosso protocolo de comunicação utiliza o auxílio de um nó intermediário (relay), que é responsável pela codificação das unidades de dados (MPDU's). Dado um cenário típico PLC, o protocolo implementado pode melhorar o *goodput* e reduzir a perda de pacotes na rede.

O ganho em vazão e resiliência na rede, a partir do uso de *network coding*, são bem conhecidos para cenários de *broadcast* em redes sem fio [Fragouli et al. 2006]. Entretanto, o uso de *network coding* ainda é pouco explorado em PLC. De fato, há poucos trabalhos na literatura que propõem o uso desse tipo de técnica em PLC [Phulpin et al. 2011, Ezzine et al. 2014]. Como principal consequência, ainda há uma falta de conhecimento sobre o comportamento de tais sistemas na prática. Por exemplo, não são bem delimitadas as circunstâncias nas quais *network coding* na camada MAC poderia melhorar o desempenho de redes PLC. Mesmo que ambos trabalhos citados demonstrem ganhos na vazão ou na resiliência da rede, pouco é tocado no que se refere à sobrecarga imposta à sinalização (ou controle) pelo uso de protocolos com *network coding*.

Para demonstrar tais melhorias, fizemos simulações de um sistema PLC utilizando estimativas mensuradas de taxa de erro de pacote. Nós utilizamos o esquema HS-OFDM (*hermitian symmetric - orthogonally frequency division multiplexing*) [Ribeiro et al. 2014] transmitindo símbolos com a modulação de chaveamento binário de fase (BPSK), por meio de canais domésticos PLC.

Em suma, nossas contribuições são duas: (i) criamos protocolos de comunicação com *network coding* em PLC. (ii) Utilizamos os protocolos desenvolvidos para delimitar os cenários nos quais o uso de *network coding* é viável e oferece melhoras no desempenho de redes PLC. Nossos resultados apontam que, em uma transmissão tipo *broadcast* e transversal, o uso de *network coding* pode aumentar o *goodput* média em até 116% em relação a um cenário tradicional. Mais ainda, a ocupação média dos *buffers* dos dispositivos de rede pode diminuir em até 112.%. Finalmente, a latência observada em um cenário que utiliza o protocolo proposto não ultrapassa 20 unidades, enquanto em um ambiente tradicional, menos de 20% dos casos respeitam esse limite.

O restante desse artigo está organizado como segue. Na seção 2, nós avaliamos o estado da arte e motivamos o presente trabalho. Na seção 3, nós descrevemos o cenário e o problema tratado. Na seção 4, nós apresentamos nossa proposta e descrevemos os protocolos desenvolvidos. A seção 5 trás as avaliações realizadas e finalmente, a seção 6

apresenta nossas principais conclusões.

2. Trabalhos Relacionados

A Internet cada vez mais experimenta tráfego gerado por aplicações em tempo real. A maioria dessas aplicações, como VoIP e vídeo conferência, são tolerantes ao atraso. É nesse cenário que os enlaces geralmente são super utilizados e, intuitivamente, *network coding* pode trazer seu máximo benefício [Prior et al. 2014]

A principal ideia por traz de *network coding* é deixar que nós intermediários no processo de comunicação misturem pacotes por operações algébricas. Assim, há uma quebra no o esquema tradicional de encaminhamento de mensagens na rede (*store-and-forward*) [Prior et al. 2014]. Por exemplo, em um cenário simples com um *relay* intermediando a comunicação entre dois nós, *network coding* possibilita a transmissão dados entre esses dois nós simultaneamente. Ou seja, o sistema virtualmente transmite as duas mensagens em um único instante de tempo [Katti et al. 2006].

O ganho em vazão e resiliência na rede, a partir do uso de *network coding*, são bem conhecidos para cenários de *broadcast* em redes sem fio [Fragouli et al. 2006]. Nesse contexto, há trabalhos que avaliam as condições de contorno do uso de *network coding* e apresentam os ganhos máximos esperados [Keshavarz-Haddad and Riedi 2014]. Porém, há poucos trabalhos na literatura que propõem o uso desse tipo de técnica em PLC [Phulpin et al. 2011, Ezzine et al. 2014, Parag and Chamberland 2010]. Mesmo que um ambiente sem fio tenha várias características em comum com PLC, os ganhos em um ambiente não podem ser inferidos automaticamente ao outro, e assim, ainda há uma falta de conhecimento sobre o comportamento de NC em PLC na prática [Parag and Chamberland 2010].

Os trabalhos existentes tentam verificar o efeito de *network coding* em PLC, na sua maior parte, em condições ideais. Eles não propõe protocolos na camada de acesso ao meio e tampouco usam cargas realistas em sua simulação. Por exemplo, [Ezzine et al. 2014] foca seu trabalho na proposta de um algoritmo de alocação de subportadoras em um sistema PLC com acesso ao meio baseado em OFDMA (Orthogonal Frequency Division Multiplexing Access). Eles avaliam os ganho de vazão na rede, quando é utilizado um mecanismo de *network coding* simples entre dois nós e um *relay*. Os autores também avaliam o sistema quando os nós estão em diferentes posições em relação ao *relay*. Apesar dos autores terem utilizado exemplos com *network coding*, esse não foi o objetivo do artigo. As avaliações elaboradas são baseadas em ganhos assintóticos, sem a consideração de um protocolo apropriado e erros realistas que podem influenciar no desempenho do sistema.

Phulpin et al. [Phulpin et al. 2011, Prior et al. 2014] também dissertam acerca de prováveis benefícios de *network coding* a uma PLC. Os autores propõem o uso de um esquema de *linear code* em um ambiente de *smart grids*, onde tanto PLC quando redes sem fio podem ser utilizado. O principal objetivo é o aumento da eficiência da coleta de dados na *smartgrid*. Entretanto, os autores também não definem um ambiente e um protocolo para o uso de *coding*. Mais ainda, não há limitações realistas no cenário descrito, como taxas de erro nos canais ou limitações no tamanho dos *buffers*.

Na linha do trabalho que propomos, Bilbao et al. [Bilbao et al. 2014] propõem um protocolo também em camada de acesso ao meio PLC que utiliza *network coding*.

Eles dedicam parte do trabalho à caracterização de erro em PLC de banda estreita e mostram resultados que indicam que *network coding* pode aumentar a vazão da rede em até (aproximadamente) 3x, se comparado uma um sistema tradicional com *stop&wait*. Apesar das semelhanças, destacamos que o ambiente experimentado é diferente do proposto nesse trabalho. Acreditamos que nesse trabalho, os autores apresentam um limite superior para os ganhos de *network coding* em PLC. Mais ainda, os autores não avaliam questões importantes, como a ocupação de *buffers* na rede e a latência total para entrega de dados.

3. Cenário avaliado

Neste trabalho, consideramos um ambiente típico de PLC, onde há uma sequência de residências (nodos) interconectadas por um canal comandando por um coordenador (nó central). Note que, cada nó pode contemplar diversos dispositivos gerando tráfego na rede. Assim como acontece em meios sem fio, alguns destes nós podem não ter contato com outros na rede, ora por estarem em fases diferente, ora por uma alta atenuação do enlace [Ferreira et al. 2010].

Neste trabalho, consideramos um esquema *uncoded* HS-OFDM juntamente com a modulação de chaveamento de fase binário (BPSK). Consideramos também perfeita sincronização e conhecimento da informação de estado do canal por parte do receptor. A potência total é de $P = P_0 + P_1$ onde P_0 e P_1 são alocadas para os nós S e R, nessa ordem. As potências são distribuídas igualmente entre as N subportadoras do símbolo HS-OFDM (P_0/N e P_1/N para os nós S e R, respectivamente), durante um ciclo de comunicação de dados (primeiro *time-slot* alocado para o nó S e o segundo, para o nó R).

Nós consideramos um cenário simples, como mostramos na Figura 1, onde a origem de dados se comunica com múltiplos destinos. Cada nó é equipado com um *buffer* de dados onde pacotes de dados são armazenados para posterior transmissão. Mais precisamente, de acordo com a topologia definida na Figura 1, temos cinco nós (A, B, C, D e E). Não há conexão completa entre esses nós, e assim, as linhas nessa figura indicam o compartilhamento do enlace entre os dispositivos PLC, caracterizando um *overhearing* perfeito (A e D podem escutar B e C; e vice-versa). O nó E, geralmente o concentrador na rede, tem capacidade de comunicação com todos os outros.

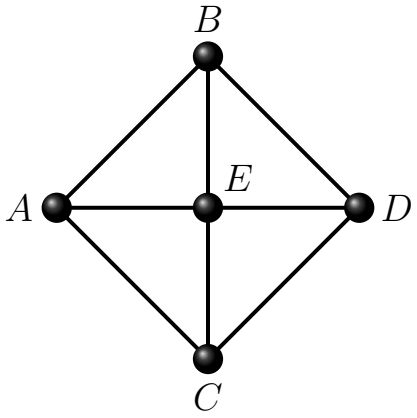


Figura 1. Topologia do cenário estudado.

Cada enlace i, onde $i \in \{AB, AC, BD, CD, AE, BE, CE, DE\}$ possui taxas de erro de pacote (PER_i) independentes. Note que, apesar de existir apenas um nó *relay*

nesse cenário, a proposta atual pode ser expandida ao uso de múltiplos *relays* de forma transparente ao protocolo proposto na seção 4. Além disso, focamos nesse trabalho em mostrar um cenário simples, onde a aplicação de *network coding* é apropriada.

Os valores das taxas de erro de pacote (PER_i) são extraídos a partir da taxa de erro de bits (BER_i) obtida na camada física para o i-ésimo enlace. Seu cálculo é dado por $PER_i = 1 - (1 - BER_i)^{N_i}$, onde N_i corresponde ao tamanho do pacote. Essa fórmula representa a probabilidade de ocorrência de erro em, pelo menos, um bit de um pacote com N_i bits. Na prática, os valores de PER_i na camada de enlace são desconhecidos.

Nós estimamos valores para erros a partir de campanhas de medição realizadas em uma área típica urbana brasileira [Roberto de Oliveira 2016], onde foram coletadas informações sobre taxas de erros de bit em mais de 36 mil canais PLC, considerando a frequência de banda entre $1,705$ MHz e 100 MHz. Neste trabalho, nós utilizamos dados das medições realizadas em canais com a potência total de 30 dBm, com taxa média de erro de 18% em cada enlace.

4. Protocolo de aplicação do *network coding* XOR em um sistema PLC

Para melhorar o desempenho da rede proposta na figura 1, podemos utilizar uma estratégia de *network coding* XOR [Katti et al. 2006], implementada na camada de controle de acesso ao meio (MAC) de um sistema PLC. Para tal, qualquer um dos nós da topologia poderia ser escolhido para executar o papel de nó codificador, responsável por armazenar em um buffer todo *MAC Protocol Data Unit* (MPDU) recebido, executar uma disjunção exclusiva em todos eles para gerar um MPDU novo que será a combinação de todos outros. O nó codificador gera uma mensagem múltipla para todos os usuários vizinhos, que tem informação suficiente para decodificar o MPDU codificado.

Contudo, espera-se obter melhor desempenho se o nó escolhido for posicionado de maneira central, tal qual o nó E da topologia apresentada na figura 1. O nó E é beneficiado por sua posição relativa aos outros nós da topologia por ter acesso físico a 1-*hop* de distância de qualquer outro nó. Outra vantagem da escolha de um posicionamento apropriado é que podemos utilizar um escalonamento de *slots* TDMA a fim de favorecer a codificação, estabelecendo o *slot* final de um frame[1] TDMA-OFDM ao nó que executará a combinação (XOR no caso desse trabalho).

Um exemplo de como *network coding* pode propiciar a economia de mensagens em um esquema *multihop* é dado na figura 2.

No exemplo, os MPDU1 e MPDU2 do primeiro quadro são codificados em uma única mensagem ($msg3$), que pode ser enviada ainda no mesmo *frame*, dentro do *time-slot* designado ao nó codificador. Com isso, em um ambiente em que sejam executadas transmissões de fluxo transversal, é possível que uma troca de mensagens sejam executadas em apenas um *frame* TDMA-OFDM, além de proporcionar a economia de uma mensagem para concretizar a comunicação. Dentro da topologia definida no problema, esperamos que a mesma situação ocorra e que os ganhos em termos de economia de mensagem se traduzam em um melhor desempenho geral do sistema.

[1]Nesse contexto, frame é o conjunto de todos as mensagens transmitidas durante todos os *slots* de tempo de uma rodada do TDMA.

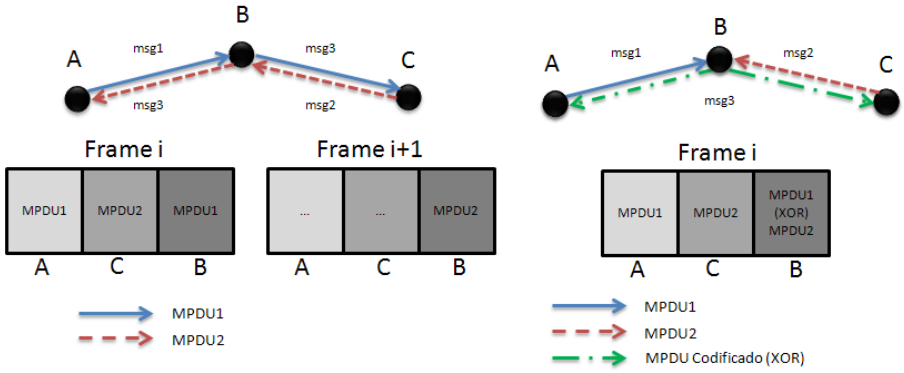


Figura 2. Exemplo de utilização de network coding em TDMA

Para a implementação do *network coding*, definimos procedimentos de transmissão e recepção, tanto para nós periféricos, quanto para o nó central. O perfil de alocação de *slots* TDMA-OFDM em um *frame* i pode ser visto na figura 3. Dentro dos *time-slots* t_i, referente a cada um dos nós u, ocorre o envio do MPDU do respectivo usuário e o recebimento do mesmo pelos usuários vizinhos ao transmissor. No *slot* de tempo de controle, são enviados ACK e NACK relativos às transmissões no *frame* anterior. Para fins de praticidade na simulação, considera-se que o período de controle é livre de erros.

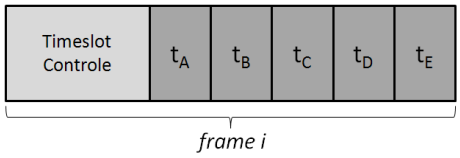


Figura 3. Perfil de alocação de um frame TDMA-OFDM

A estrutura do MPDU pode ser vista na figura 4. Nela, são definidos campos de origem e destino da mensagem, o tipo (dados, controle ou mensagem codificada) e em qual *frame* essa mensagem foi recebida pela camada MAC. O campo ”componentes”é um *bitmap* que define quantos e quais usuários estarão presentes em um MPDU codificado. Como utilizamos um esquema TDMA-OFDM de *timeslot* fixo, cada bit do campo representa, em ordem, um *timeslot* do respectivo usuário.

Figura 4. Estrutura de MPDU

Os algoritmos que controlam os processos de transmissão e recepção dos nós do sistema simulado são descritos nos fluxogramas das figuras 5 e 6. A figura 5 descreve os procedimentos de transmissão, recepção e decodificação de MPDU's um nó u em seu respectivo *time-slot*. Quando um erro de MPDU ocorre, o algoritmo compreende também o processo de retransmissão, que está inserido no *time-slot* de controle. Na figura

6, o fluxograma descreve os procedimentos de recepção, transmissão e codificação de MPDU's pelo nó responsável por executar o *network coding* PLC.

Mais precisamente, dentro do algoritmo proposto para o *network coding* PLC, no *time-slot* correspondente, o nó atual executará os seguintes procedimentos:

- Análise de MPDU de controle do último *time-slot*: em um *frame* i, no *time-slot* de controle, um nó u recebe um mensagem de ACK/NACK de seus respectivos vizinhos. Se ACK, a mensagem no *frame* $i-1$ foi corretamente enviada. Se NACK, a mensagem do *frame* $i-1$ é preparada para ser retransmitida no *frame* i, dentro do *time-slot* relativo ao nó u.
- Recepção de MPDU de dados: em cada *time-slot* t_u de um *frame* i, um nó u recebe o MSDU dos seus vizinhos. Se o MPDU foi corretamente recebido, ele é armazenado em um buffer de recepção e uma mensagem de ACK é preparada para ser enviada a todos os vizinhos, no *time-slot* de controle do *frame* $i+1$. Se a mensagem tem erro, um NACK é enviado para todos os vizinhos e os MSDU's recebidos no *frame* i são descartados.
- Recepção de MPDU codificado: em um *frame* i, no último *time-slot*, todo nó u recebe uma mensagem codificada do nó responsável pelo *network coding*, contendo as mensagens corretamente recebidas e ele codificadas dentro de um MSDU codificado. Em casos de erro no envio do MPDU codificado, no *time-slot* de controle do *frame* $i+1$, o nó u envia mensagem de NACK.
- Decodificação de MPDU codificado, caso seja possível: para a decodificação de um MPDU codificado em u, é necessário que as mensagens dos vizinhos que chegaram sem erro ao nó de codificação, também sejam corretamente recebidas por ele. Se essas mensagens estiverem no *buffer* de recepção de um nó u, esse nó consegue extrair o MPDU que é destinado a ele. Caso os dados de decodificação não estiverem presentes no *frame* atual, o MPDU codificado é armazenado em *buffer* até que todas as mensagens do *frame* anterior sejam retransmitidas.
- Codificação com disjunção exclusiva: em todo *frame* i, o último *time-slot* t é designado ao usuário E, que executa a codificação de todos os MPDU's corretamente transmitidos pelos demais usuários no *frame* e transmite em múltiplas mensagens para todos usuários.
- Transmissão de MPDU gerado pela camada superior: quando um dado é gerada na camada superior de um usuário u, um MPDU é criado e armazenado na fila de transmissão. Ao ser transmitido, em um *frame* i, o MPDU fica armazenado nesse *buffer* até a recepção do ACK ou NACK no *time-slot* de controle do frame $i+1$.

5. Avaliação

5.1. Metodologia

Para analisar o desempenho do sistema gerados pelos algoritmos propostos, criamos, no *software* MATLAB, um simulador da topologia definida na figura 1. Nesse sentido, foram testados dois cenários:

- Rede *multihop* com todos os cinco nós iguais. A camada de enlace é responsável por transportar MPDU's utilizando um nó intermediário aleatório entre transmissor e destino, fazendo com que essas MPDU's alcancem usuários que não são acessíveis a 1-*hop* pelo nó de origem da transmissão. Todos nós transmitem para

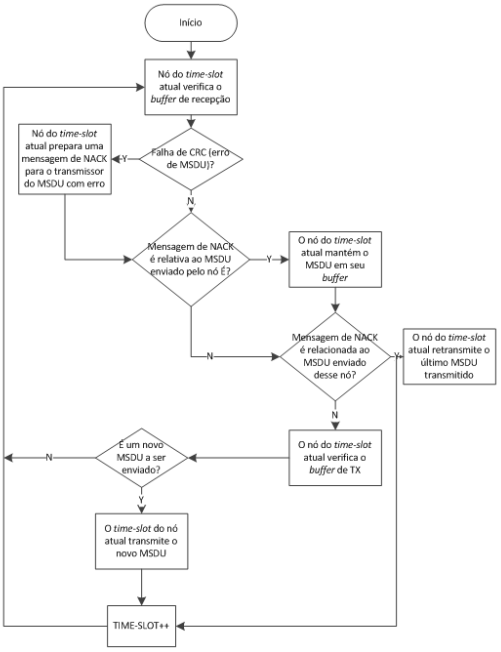


Figura 5. Procedimento de recepção e transmissão com erro no MPDU

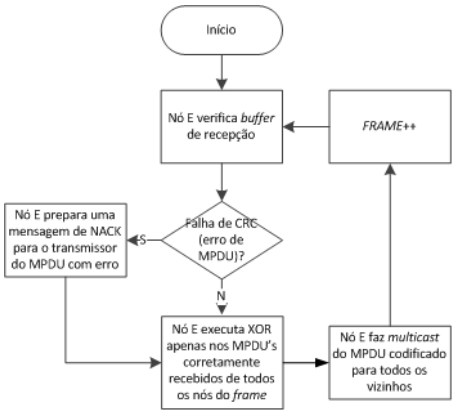


Figura 6. Procedimento de codificação, recepção e transmissão em um nó de Network Coding

todos outros em *multicast*. Assume-se que, para todo nó (exceto o central), há uma probabilidade de geração de uma MPDU de tamanho fixo a ser transmitido em cada *frame* TDMA-OFDM (ou seja, um nó transmite com uma dada probabilidade).

- Rede *multihop* com 4 nós iguais e o nó E (central) responsável pelo *network coding*. O nó central está sempre disponível para receber MPDU's de seus vizinhos, codificá-los e colocá-los em uma fila de transmissão. Assume-se que, para todo nó, há uma probabilidade de geração de uma MPDU de tamanho fixo a ser transmitido em cada *frame* TDMA-OFDM.

Nesses cenários, dois parâmetros foram variados: o tamanho das MPDU's (40, 80, 120, 160, 200 e 248 Bytes) e a probabilidade de um novo MPDU ser gerado pela camada superior e adicionado à fila de transmissão (25, 50 e 75%). Essas variações são interes-

santes pois impactam na taxa de erro de pacotes (i.e. tamanho da MPDU) e o tráfego de dados no sistema simulado (i.e. taxa de transmissão). Os tamanhos utilizados são baseados em estudo empírico que demonstra que mais da metade dos pacotes que trafegam na Internet tem tamanho menor que 300 Bytes [Sinha et al. 2007]. As probabilidades de transmissão apresentam um bom compromisso para demonstrar o comportamento da rede em baixa, média e alta carga de trabalho.

Conforme definimos na seção 3, consideramos também um sistema *uncoded* TDMA-OFDM, que aloca todas as subportadoras para um nó quando o mesmo está usando seu próprio *time-slot* para transmissão. Adotamos a modulação digital BPSK e uma potência $P = 30$ dBm. A frequência de banda utilizada é de 1,7 até 100 MHz. O tráfego na rede simulada é totalmente transversal (ou seja, A e B enviam pacotes destinados a C e D, e vice-versa).

Em cada cenário avaliado, e em cada uma de suas variações, foram executados os mesmos experimentos, a fim de apurar os seguintes valores: ocupação média dos *buffers*, *goodput* e latência média. A ocupação média de *buffers* é definida pelo total de MPDU's válidas no *buffers* de todos usuários do sistema em um determinado intervalo de tempo. O *goodput* é e definido pelo número de recepções corretas por um nó u em um dado período de tempo. A latência média foi dada pela média do número de *frames* TDMA-OFDM que um MPDU levou para ser entregue corretamente em seu destino, em um determinado intervalo de tempo.

Para medições de ocupação de *buffers*, foram realizadas 1.000 iterações, onde cada uma corresponde a 100 *frames* TDMA-OFDM. Para medições de *goodput*, fora realizadas 1.000 iterações, onde cada uma corresponde à chegada correta de 100 MPDU's a um nó especificado. No caso da medição de latência média, foram realizadas 1.000 iterações, onde cada uma compreende os envios realizados durante a execução de 200 *frames* TDMA-OFDM.

5.2. Resultados numéricos

A figura 7 mostra a distribuição acumulada (CDF) da relação entre a ocupação total dos *buffers* em cada *frame* TDMA-OFDM, em um cenário *multihop* comum e o cenário *network coding* PLC. A taxa de geração de MPDU's na camada superior utilizada nesse foi de 50% (rede com carga média de trabalho). Nesse caso, os testes foram executados com MPDU de tamanho com tamanho fixo de 248 Bytes.

Note que, há situações que a rede sem NC (cenário comum) pode apresentar valores de ocupação de *buffer* menores que um cenário com NC. Intuitivamente, quando o sistema tem pouca disputa por recursos, a entrega direta (no caso, por 2 hops) pode gerar menos ocupação dos *buffers* do sistema. Entretanto, para cerca de 90% dos casos, o cenário com NC apresentou melhor desempenho. Observe que em 50% dos casos, uma rede comum tem ocupação de buffer média acimda da máxima ocupação média encontrada em um cenário com NC. De forma geral, a diminuição média de ocupação de *buffer* pelo uso de NC é alta, chegando a 200% de diferença mediana.

Essa melhoria está relacionada ao fato de, ao adotarmos o protocolo proposto, desobrigamos todos os nós do sistema de exercer um papel de intermediário entre a comunicação de dois nós *multihop*. Com isso, evitamos gargalos que teriam que ser tratados com algoritmos de roteamento mais sofisticados e com maior *overhead* de mensagens

de controle. Como o algoritmo utilizado para o *network coding* é o XOR, o aumento de tamanho do MPDU se dá apenas com alguns campos de controle. Outro fator importante que contribui para o menor uso de *buffers* é o algoritmo utilizado na decodificação dos MPDU's. Quando um nó vizinho não tem os dados necessários para a decodificação de um MPDU codificado, ele descarta os MPDU's do *frame* anterior, conserva apenas mensagem codificada e pede reenvio a todos os vizinhos (menos ao nó de *coding*).

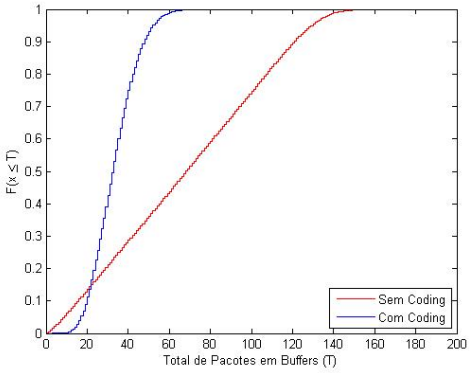


Figura 7. Ocupação total de *buffers* do sistema ao fim de 100 *frames*.

Na figura 8, é apresentada a função de distribuição acumulada da relação do *goodput* de todos os usuários do sistema entre os cenários testados. Como esperado, há são necessários menos *frames* TDMA-OFDM para realizar a entrega das MPDU's. Como consequências, a vazão média do sistema se eleva. Mais ainda, há uma redução das filas dos buffers (fig. 7) e assim, menos MPDU's ficam retidos em filas de *buffers* de usuários intermediário. Esses dois fatores contribuem para o ganho observado no gráfico. Na mediana, a diferença é de cerca de 40% entre os dois cenários.

É interessante observar que, utilizando outros algoritmos para regular o processo de ACK e NACK dos pacotes a serem utilizados na decodificação, seria possível ter um ganho ainda maior em *goodput*, em troca de mensagens de controle e uso de *buffers* para armazenar mensagens passadas. Em outras palavras, o nosso algoritmo, de maneira simplista, pede retransmissão de todos os MPDU's em um *frame* que houve erro. Caso fosse implementado mecanismos de retransmissão seletiva, o *goodput* poderia ser ainda melhor em uma rede PLC com *network coding*.

A figura 9 apresenta o *goodput* médio dos nós sistema, em relação a diferentes tamanhos de MPDU. Para os dados apresentados, o intervalo de confiança, para 99% de confiança, é desprezível (menor que $0,01\%$) Em ambos os cenários, observamos um leve decaimento na métrica de interesse ao se elevar o tamanho das mensagens. Para todas as variações, a diferença observada entre os sistemas é de cerca de 41%. O impacto pelo crescimento do tamanho da MPDU pode estar relacionado a um leve aumento na taxa de erro de pacotes que nesse caso, influencia negativamente ambos os cenários, gerando mais retransmissões.

A variação da taxa de transmissão (probabilidade de se transmitir dados) destaca ainda mais a diferença de desempenho entre os sistemas avaliados. A figura 10 apresenta os valores médios de *goodput* ao se variar essa taxa. Novamente, os valores apresentados são médias e o intervalo de confiança é desprezível (com nível de confiança de 99% é des-

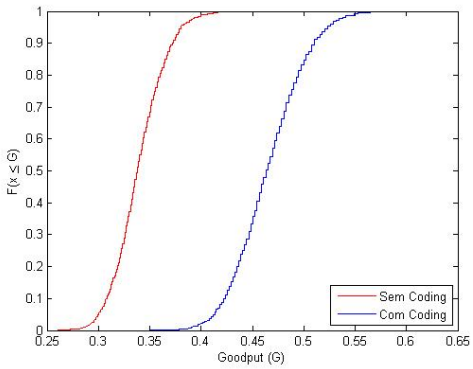


Figura 8. *Goodput* total dos nó do sistema ao fim de 100 MPDU's corretamente recebidas.

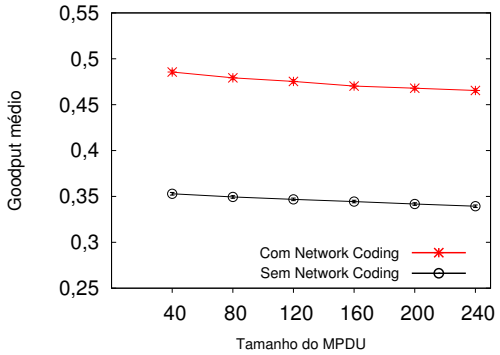


Figura 9. Média do *goodput* do sistema em relação à variação de tamanho de MPDU.

prezível). Por essa figura, observamos que um sistema sem NC satura mais rapidamente. Enquanto o *goodput* ainda é crescente quando há 75% de chances de transmissão com um sistema em NC, a diferença entre os dois últimos pontos para um sistema sem NC é pouco significante. Com a rede em baixa carga, a diferença entre os 2 sistemas também é baixo. Porém, com 75% de prob. de transmissão em cada nó, a diferença média alcança mais de 115%.

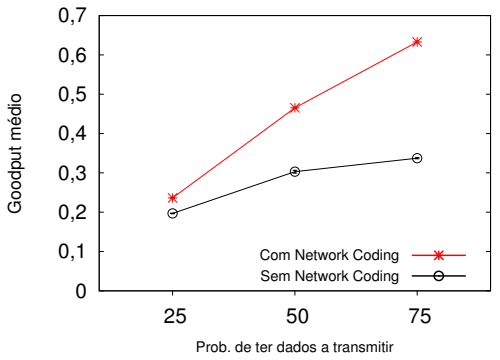


Figura 10. Média do *goodput* do sistema em relação à probabilidade de geração de um MPDU na camada superior.

As figuras 11 e 12 apresentam a ocupação de buffer no sistema, ao se variar o ta-

manho das MPDU's e a taxa de transmissão, respectivamente. Assim como para goodput, o aumento do tamanho da MPDU também piora o desempenho de ambos os sistemas. Como a taxa de erros de pacotes é proporcional ao tamanho deles, um aumento na MPDU aumenta a taxa de erro do sistema, o que gera mais retransmissões (e filas nos *buffers* dos dispositivos). Em redes com baixa carga, a transmissão direta, sem necessidade de codificação/decodificação apresenta vantagens em relação ao uso de NC. Esse comportamento afirma o resultado apresentado pela figura 7. Entretando, com o aumento da taxa de transmissão (i.e. maior probabilidade de transmissões), a rede se torna mais disputada pelos dispositivos. Nesse cenário, o sistema com *network coding* sofre menos impacto que o sistema tradicional. A diferença de desempenho entre os dois sistemas em alta carga chega a ser de aproximadamente 112%.

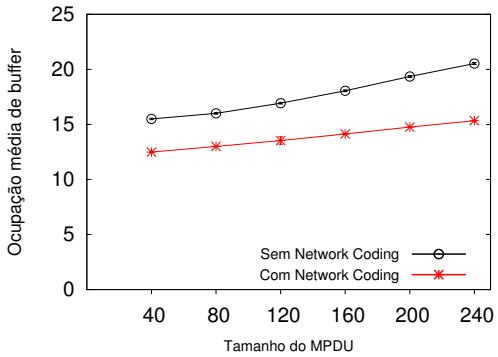


Figura 11. Média de ocupação de *buffer* no sistema em relação à variação de tamanho de MPDU.

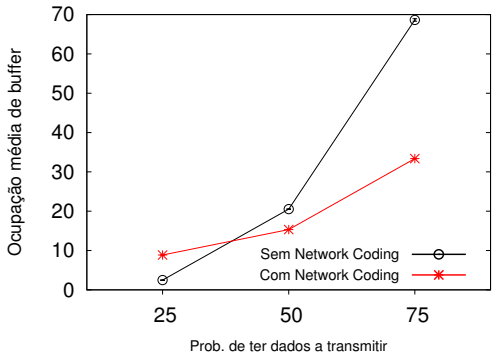


Figura 12. Média de ocupação de *buffer* no sistema em relação à probabilidade de geração de um MPDU na camada superior.

Finalmente, o uso do protocolo proposto com NC proporciona transmissões com menos latência. De fato, virtualmente há um paralelismo nas transmissões, conforme exemplificamos na sessão anterior. De acordo com a figura 13, enquanto a latência máxima experimentada em um sistema com coding foi de apenas 10 unidades (execuções de frame), em sistemas tradicionais, apenas 20% das transmissões conseguiram atingir esse limite. Em mais de 50% dos casos, a latência foi superior a 20 unidades.

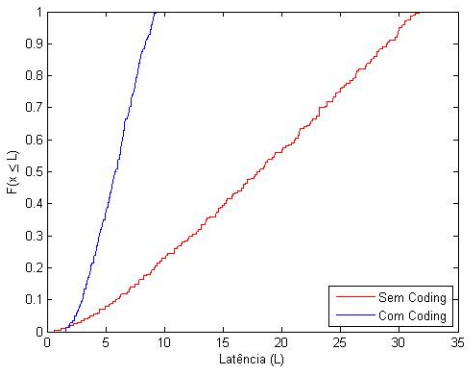


Figura 13. Latência média dos MPDU's em um determinado intervalo de tempo.

6. Conclusões

Nesse artigo, nós propomos um protocolo de comunicação com uso de *network coding* camada MAC para redes PLC que faz uso de um modelo com um único *relay*. Nós propomos o desempenho do protocolo a partir de simulações de uma topologia comum em um sistema PLC com acesso múltiplo por divisão no tempo (TDMA-OFDM).

Nossas simulações mostram ganhos expressivos com a utilização do novo protocolo. Nesse caso, novo protocolo é capaz de reduzir a perda de pacotes e aumentar o *goodput*. De fato, comparado a um sistema tradicional baseado em *stop&wait*, um sistema utilizando o protocolo proposto pode atingir um *goodput* até 116% maior. Mais ainda, a taxa de ocupação de *buffers* no sistema cai pela metade. Finalmente, a latência observada com o uso do novo protocolo, na mediana é 4 vezes menor.

Trabalhos futuros incluem extensões para novos protocolos, especialmente em ambientes híbridos sem-fio/PLC. Mais ainda, pretendemo agregar o uso de network coding com cooperação oportunistas nesses sistemas, reduzindo assim a necessidade de um *relay* específico.

7. Agradecimentos

Os autores agradecem o apoio do CNPq, CAPES e da FAPEMIG.

Referências

Biagi, M. (2011). Mimo self-interference mitigation effects on power line relay networks. *IEEE Communications Letters*, 15(8):866–868.

Bilbao, J., Calvo, A., Armendariz, I., Crespo, P. M., and Medard, M. (2014). Reliable communications with network coding in narrowband powerline channel. In *Proc. IEEE International Symposium on Power Line Communications and its Applications*, pages 316–321.

Cheng, X., Cao, R., and Yang, L. (2013). Relay-aided amplify-and-forward powerline communications. *IEEE Trans. on Smart Grid*, 4(1):265–272.

Ezzine, S., Abdelkefi, F., Cances, J. P., Meghdadi, V., and Bouallegue, A. (2014). Joint network coding and ofdma based mac-layer in plc networks. In *Power Line Commu-*

nications and its Applications (ISPLC), 2014 18th IEEE International Symposium on, pages 311–315. IEEE.

Ferreira, H. C., Lampe, L., and Newbury, J. (2010). *Power Line Communications Theory and Applications for Narrowband and Broadband Communications over Power Lines*. John Wiley Sons, United Kingdom, 1st edition.

Fragouli, C., Le Boudec, J.-Y., and Widmer, J. (2006). Network coding: an instant primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68.

Gao, Q., Yu, J., Chong, P., So, P., and Gunawan, E. (2008). Solutions for the "silent node" problem in an automatic meter reading system using power-line communications. *Power Delivery, IEEE Transactions on*, 23(1):150–156.

Katti, S., Rahul, H., Hu, W., Katabi, D., Médard, M., and Crowcroft, J. (2006). Xors in the air: practical wireless network coding. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 243–254. ACM.

Keshavarz-Haddad, A. and Riedi, R. H. (2014). Bounds on the benefit of network coding for wireless multicast and unicast. *Mobile Computing, IEEE Transactions on*, 13(1):102–115.

Kim, Y., Choi, S., Kim, S., and Lee, J. (2012). Capacity of ofdm two-hop relaying systems for medium-voltage power-line access networks. *IEEE Trans. on Power Delivery*, 27(2):886–894.

Noori, M. and Lampe, L. (2013). Improving data rate in relay-aided power line communications using network coding. In *Proc. IEEE Global Communications Conference*, pages 2975–2980.

Parag, P. and Chamberland, J.-F. (2010). Queueing analysis of a butterfly network for comparing network coding to classical routing. *Information Theory, IEEE Transactions on*, 56(4):1890–1908.

Phulpin, Y., Barros, J., and Lucani, D. (2011). Network coding in smart grids. In *In. proc. IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 49–54.

Prior, R., Lucani, D. E., Phulpin, Y., Nistor, M., and Barros, J. (2014). Network coding protocols for smart grid communications. *IEEE Transactions on Smart Grid*, 5(3):1523–1531.

Ribeiro, M. V., Colen, G. R., Campos, F. V. P., Quan, Z., and Poor, H. V. (2014). Clustered-orthogonal frequency-division multiplexing for power line communication: When can it be beneficial? *IET Communications*, 8(13):2336–2347.

Roberto de Oliveira, Moisés Ribeiro, A. B. V. (2016). Um protocolo mac de cooperação para redes plc. *Elsevier Computer Networks*.

Sinha, R., Papadopoulos, C., and Heidemann, J. (2007). Internet packet size distributions: Some observations. `https://www.isi.edu/~johnh/PAPERS/Sinha07a.pdf`. Acessado em: 26-12-2015.

Valencia, J., Oliveira, T. R., and Ribeiro, M. V. (2014). Cooperative power line communication: Analysis of brazilian in-home channels. In *Proc. IEEE International Symposium on Power Line Communications and its Applications*, pages 301–305.

FTSP+: A MAC Timestamp independent flooding time synchronization protocol

Heder Dorneles Soares[1], **Raphael Pereira de Oliveira Guerra**[1], **Célio Vinicius Neves de Albuquerque**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Av. Gal. Milton Tavares de Souza, São Domingos – Niterói – RJ – Brasil

{hdorneles,rguerra,celio}@ic.uff.br

Abstract. *Wireless sensors networks are distributed system composed of battery-powered nodes with low computational resources. Like in many distributed systems, some applications of WSN require time synchronization among their nodes. In the particular case of WSN, synchronization algorithms must respect the nodes computational constraints. The well known FTSP protocol is famous for achieving nanosecond precision with low overhead. However, it relies on MAC timestamp, a feature not available in all hardware. In this work, we propose MAC timestamp independent version in order to extend and adapt FTSP to work on hardware that do not have MAC timestamp while keeping the low overhead and high synchronization precision. Our results we estimate an average synchronization error of* $1.508\mu s$ *per hop, while adding a corretion message.*

1. Introduction

Wireless Sensor Networks (WSNs) are composed of small computational devices equipped with an antenna for wireless communication, one or more kind of sensors, and a small CPU for simple data processing [Baronti et al. 2007]. These devices are usually called *motes*. Due to the limited radio signal range and energetic constraint, WSNs have restrictions and unique characteristics that differ from traditional networks and distributed systems [Arampatzis et al. 2005].

WSNs have several applications in many diverse fields. Sensors deployment in military applications have always been widespread, so the introduction of motes was a natural incorporation to the advancement of systems already used. Applications enhanced with WSN include tracking enemy and targets [Yang and Sikdar 2003], monitoring of vehicles [Sinopoli et al. 2003], countersniper system [Simon et al. 2004], and surveillance systems [Gui and Mohapatra 2004]. Environmental monitoring also provide opportunities to apply WSN. Our environment has a lot of information that play an important role in our quality of life, such as quality of air, water, sound and solar radiation to which we are exposed directly and affect our health [Oliveira and Rodrigues 2011, Cardell-Oliver et al. 2004]. Increasing interest in green computing has led to concerns with energy consumption of IT facilities [Chong et al. 2014], and WSNs play a strategic role in monitoring and controlling these environments [Zanatta et al.].

Some of those applications require time synchronization mechanisms with good accuracy and scalability, all this complying with their low computational resources and

energy availability [Zanatta et al. , Simon et al. 2004, Yang and Sikdar 2003]. Flooding Time Synchronization Protocol (FTSP) is the most popular time synchronization algorithm for WSN [Maróti et al. 2004]. It is fault-tolerant, achieves high accuracy ($\sim 1,5\mu s$ per hop) utilizing timestamps in low layers of the radio stack, and saves energy using a linear regression technique to compensate clock skews with few exchanges of synchronization messages. However, MAC layer timestamping is not a standardized feature, and hence, not interoperable among different hardware and physical layer protocols. There is an effort from Google to standardize MAC layer timestamping in WSN [Wang and Ahn 2009], but so far there is not much compliance.

Many synchronization protocols use MAC timestamp: some have less accuracy than FTSP, focus at other problems and make more restrictive assumptions [Ganeriwal et al. 2003, van Greunen and Rabaey 2003]; others can achieve better accuracy between distant nodes [Nazemi Gelyan et al. 2007, Lenzen et al. 2009, Sommer and Wattenhofer 2009]. Elson et al. proposed the Reference Broadcast Synchronization [Elson et al. 2002] (RBS) to eliminate uncertainty of the sender without MAC timestamp by removing the sender from the critical path. The idea is that a third party will broadcast a beacon to all receivers. The beacon does not contain any timing information; instead the receivers will compare their clocks to one another to calculate their relative phase offsets. It has $\sim 30\mu s$ error per hop and is independent of MAC timestamp. Ranganathan and Nygard offer a good overview of these protocols [Ranganathan and Nygard 2010].

In this paper, we propose a modified version of FTSP, called FTSP+, in order to work without MAC timestamp. For that, we use application level time-stamping on synchronization messages, which leads to a well-known problem: the time elapsed between time-stamping the packet and gaining the wireless medium reduces synchronization accuracy. We circumvent this problem using medium access interrupt handlers on the sender to measure the time needed to gain the medium, and correction messages to reduce sender side time uncertainty.

Our experimental results show that we only double FTPS synchronization inaccuracy, a great result compared to RBS which is 10 times worse. We also present a quantitative evaluation of medium access and processing delays on TinyOS, an evaluation that we have not found in related work.

There rest of this paper is organized as follows. Section 2 briefly recalls FTSP as needed for this work. Section 3 describes FTSP+. Section 4 brings our experiment results, and finally, Section 5 brings the concluding remarks.

2. Flooding Time Synchronization Protocol

This section briefly describes the Flooding Time Synchronization Protocol (FTSP). For more detailed information, refer to [Maróti et al. 2004].

FTSP is a synchronization protocol for WSN that provides high accuracy, consumes few resources, uses little bandwidth and is fault-tolerant. It elects a node as root to provide the time reference for synchronization; if root failure is detected (using timeouts), another root is elected. Root and synchronized nodes send synchronization messages periodically, and receiving nodes use these messages to synchronize. Therefore, FTSP supports multi-hop networks.

Synchronization messages comprise a *sender timestamp* which is the estimated global time and *rootID* which is the network identifier of the root (where the node with the lowest ID is the chosen root). *seqNum* is a sequence counter that is incremented each synchronization round; this field is used to verify the redundancy of messages [Maróti et al. 2004].

All nodes think they are root when the network starts, so they broadcast synchronization messages to the network. When they receive a synchronization message, they check who has the lowest ID: if the local ID is higher, this node gives up on being root and starts synchronizing. Another important check is the *seqNum*. If it is greater than the local value *highestSeqNum*, it means that this is a new synchronization message and starts the synchronization procedure.

The synchronization procedure consists of computing a linear regression [Elson and Estrin 2003] that will provide the clock skew (used to estimate the global time) in relation to the reference node. The last step is to forward its local (synchronized) time to other nodes.

```
1  event Radio.receive(TimeSyncMsg *msg){
2    if( msg->rootID < myRootID )
3        myRootID = msg->rootID;
4    else if( msg->rootID > myRootID
5      || msg->seqNum <= highestSeqNum )
6        return;
7    highestSeqNum = msg->seqNum;
8    if( myRootID < myID )
9        heartBeats = 0;
10   if( numEntries >= NUMENTRIES_LIMIT
11     && getError(msg) > TIME_ERROR_LIMIT )
12       clearRegressionTable();
13   else
14       addEntryAndEstimateDrift(msg);
15 }
```

Figure 1. FTSP Receive Routine [Maróti et al. 2004]

Figure 1 shows the routine of receiving synchronization messages. Lines 2 and 3 compare the *rootID* of the synchronization message with the local *rootID*. If the message has a lower *rootID*, the node assumes this *rootID* as root. Lines 4 to 7 ignore messages with higher *rootID* and lower *seqNum*. If *seqNum* is higher, local *highestSeqNum* is updated. Lines 8 and 9 makes a root node give up on being root when it has a *rootID* lower than its own ID. In case the *rootID* is larger it is checked whether the *seqNum* is greater or equal to the value of *highestSeqNum*, this check prevents information redundancy because the message will only be used when the *rootID* is less than or equal to *myRootID* and the number greater than the value of *highestSeqNum*. Lines 10 to 15 verify if the time of a message is in disagreement with the earlier estimates of global time, if applicable clear the regression table, if not, accumulate synchronization messages to calculate the linear regression and synchronize.

```
1  event Timer.fired() {
2    ++heartBeats;
3    if( myRootID != myID
4      && heartBeats >= ROOT_TIMEOUT )
5        myRootID = myID;
6    if( numEntries >= NUMENTRIES_LIMIT
7      || myRootID == myID ){
8        msg.rootID = myRootID;
9        msg.seqNum = highestSeqNum;
10       Radio.send(msg);
11   if( myRootID == myID )
12       ++highestSeqNum;
13   }
14 }
```

Figure 2. FTSP Send Routine [Maróti et al. 2004]

Figure 2 shows the sending routine. A node decides to be root because has not received a synchronization message for ROOT_TIMEOUT (lines 3 to 5). A node sends synchronization messages if it is root or has synchronized (lines 6 to 10). If a node is root, it also has to increment its *highestSeqNum*.

3. FTSP+

In any distributed time synchronization technique, nodes have to tell each other their local time. Figure 3 depicts this scenario. The sending node stores its local time $t1'$ in the synchronization message at time $t1$ and orders the message to be sent. Due to medium random access uncertainties, the sending node only gets access to the medium at time $t2$ and starts sending me message. $t2 - t1$ is called *medium access time*. The message propagates over the medium for an interval of time tp until it reaches the receiver radio at time $t3$. tp is the *propagation time*. Due to interrupt handling policies and packet header processing, the receiver node timestamps the message receipt at time $t4$ with its local time $t4''$. $t4 - t3$ is the *processing time*.

Synchronization inaccuracy happens because the receiving node thinks that at time $t4$ the sender has time $t1'$ and the receiver has time $t4''$. We can see in Figure 3 that this is not true. At time $t4$, the sending node has time $t4' = t1' +$ *medium access time* $+$ *propagation time* $+$ *processing time*. MAC layer time-stamping makes *medium access time* and *processing time* equal zero. Since *propagation time* is negligible ($\sim 1\mu s$) [Maróti et al. 2004], some synchronization policies — including FTSP — can achieve very good accuracy. However, without MAC layer time-stamping, these times are non-negligible and have to be calculated.

FTSP+ calculates *medium access time* using an interrupt handler to time stamp the moment that the node gets medium access to send the synchronization message. Although medium access is granted at time $t2'$, *medium access timestamp* equals $t2' + \delta$, where δ is the overhead to process the interrupt handler.

The sender sends a correction message with content $t2' + \delta - t1'$ so the receiver

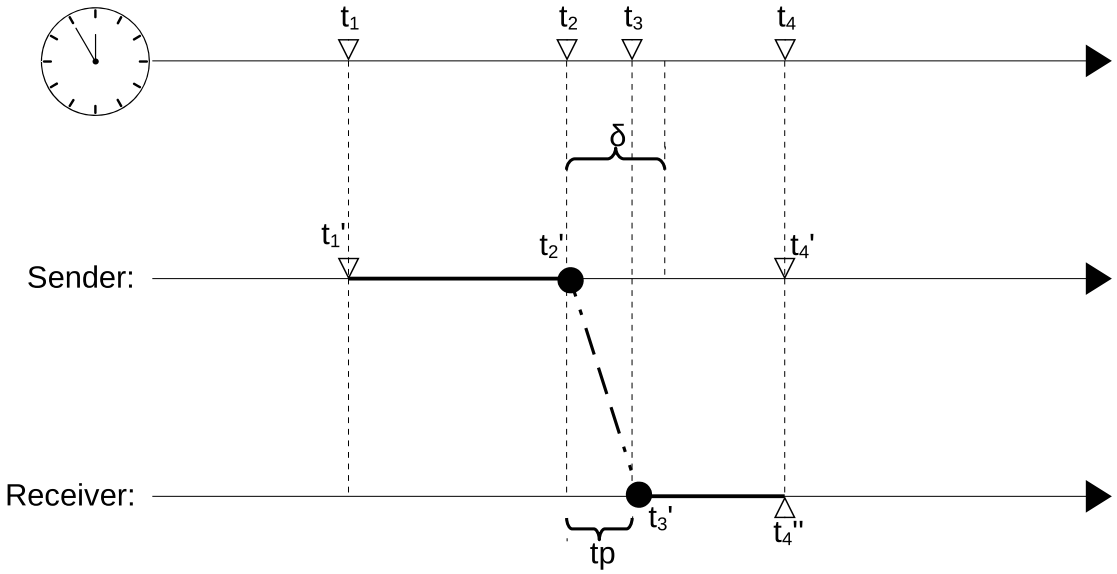


Figure 3. Syncronization steps.

can estimate $t4'$. Let us call this estimation $t\bar{4}'$. The receiver calculates $t\bar{4}'$ as in Equation (1).

$$\begin{aligned} t\bar{4}' &= t1' - (t2' + \delta - t1') \\ t\bar{4}' &= t1' + \textit{medium access time} + \delta \end{aligned} \tag{1}$$

The difference between $t4'$ and $t\bar{4}'$, which is the estimation error, is:

$$t4' - t\bar{4}' = \textit{propagation time} + \textit{processing time} - \delta \tag{2}$$

Since *processing time* and δ are interrupt handler processing latencies, they tend to cancel out each other. We investigate their values in our experiments in Section 4. Remember that *propagation time* is negligible.

As can be seen in Figure 4, an FTSP+ receiver has routines to handle two different types of income messages: `Radio.receiveSyncMsg(SyncMsg *msg)` handles synchronization messages and `Radio.receiveCorrectionMsg(CorrectionMsg *msg)` handles correction messages. The *correction time*, expressed in Equation (3), is the information that is transmitted to the receiver to apply the correction to previous messages.

$$\textit{correction time} = t2' - t1' + \delta \tag{3}$$

The `Radio.receiveSyncMsg(SyncMsg *msg)` differs from FTSP receive only for lines 13 and 14. These lines store synchronization messages in a list to wait for the correction messages if MAC layer time-stamping is not available.

```
event Radio.receiveSyncMsg(SyncMsg *msg){
  if( msg->rootID < myRootID )
      myRootID = msg->rootID;
  else if( msg->rootID > myRootID
    || msg->seqNum <= highestSeqNum )
      return;
  highestSeqNum = msg->seqNum;
  if( myRootID < myID )
      heartBeats = 0;
  if( numEntries >= NUMENTRIES_LIMIT
    && getError(msg) > TIME_ERROR_LIMIT )
      clearRegressionTable();
  else if (MAC_Time==false){
      addToWaitCorrectionMsgList(msg);
  }else{
      addEntryAndEstimateDrift(msg);
  }
}

event Radio.receiveCorrectionMsg(CorrectionMsg *msg){
  applyCorrection(msg);
  addEntryAndEstimateDrift(msg);
}
```

Figure 4. Receiver algorithm

The event `Radio.receiveCorrectionMsg(CorrectionMsg *msg)` receives the message with the correction value, use the function `applyCorrection(msg)` that applies the correction and removes from the wait list the synchronization message that matches the incoming correction message, corrects the timestamp and calls function `addEntryAndEstimateDrift(msg)`.

```
1  event Timer.fired() {
2    ++heartBeats;
3    if( myRootID != myID
4      && heartBeats >= ROOT_TIMEOUT )
5        myRootID = myID;
6    if( numEntries >= NUMENTRIES_LIMIT
7      || myRootID == myID ){
8        msg.rootID = myRootID;
9        msg.seqNum = highestSeqNum;
10       initialTime = call getLocalTime();
11       msg.timestamp = initialTime;
12       Radio.send(msg);
13   if( myRootID == myID )
14       ++highestSeqNum;
15   }
16 }
17
18 event sendDone(SyncMsg *msg, error_t error){
19    finalTime = call getLocalTime();
20    correctionMsg.correction = finalTime-initialTime;
21    correctionMsg.seqNum = msg.seqNum;
22    Radio.send(correctionMsg);
23 }
```

Figure 5. Sender algorithm

As can be seen in Figure 5, an FTSP+ sender has two routines: `Timer.fired()` periodically sends synchronization messages (like in FTSP) and `sendDone(message_t *msg, error_t error)` sends the correction message.

The function `Timer.fired()` differs from FTSP only for lines 10 and 11, which collects the timestamp at application level. `sendDone(message_t *msg, error_t error)` handles the interrupt when wireless medium access is granted to collect the medium access timestamp by compute the time that has passed between `send/sendDone` and send out the correction message, this technique is previously discussed by [Sousa et al. 2014] that is a simple technique for local delay estimation in WSN.

Lines 20-21 show how the correction time is computed. `seqNum` is used to identify which message is being adjusted, `finalTime` and `initialTime` refers to times $t2$ and $t1$ in Equation (3).

4. Evaluation

In this section, we describe our experiments to assess FTSP+'s accuracy. We have implemented FTSP+ on TinyOS 2.1.2 [Levis et al. 2005] and ran our tests on Micaz motes [MEMSIC 2015]. Micaz motes do support MAC layer timestamping, but we need this information to measure our correction accuracy. For synchronization, we overwrite MAC timestamp with our application layer timestamp.

We use *jiffys* as time unit because this is the time basis of TinyOS and represents the time between 2 clock ticks. Our experiments results correspond to 10 minutes experiments with synchronization messages being sent every 3 seconds, with 2 nodes communicating and a base station connected to an computer via serial port for record the messages.

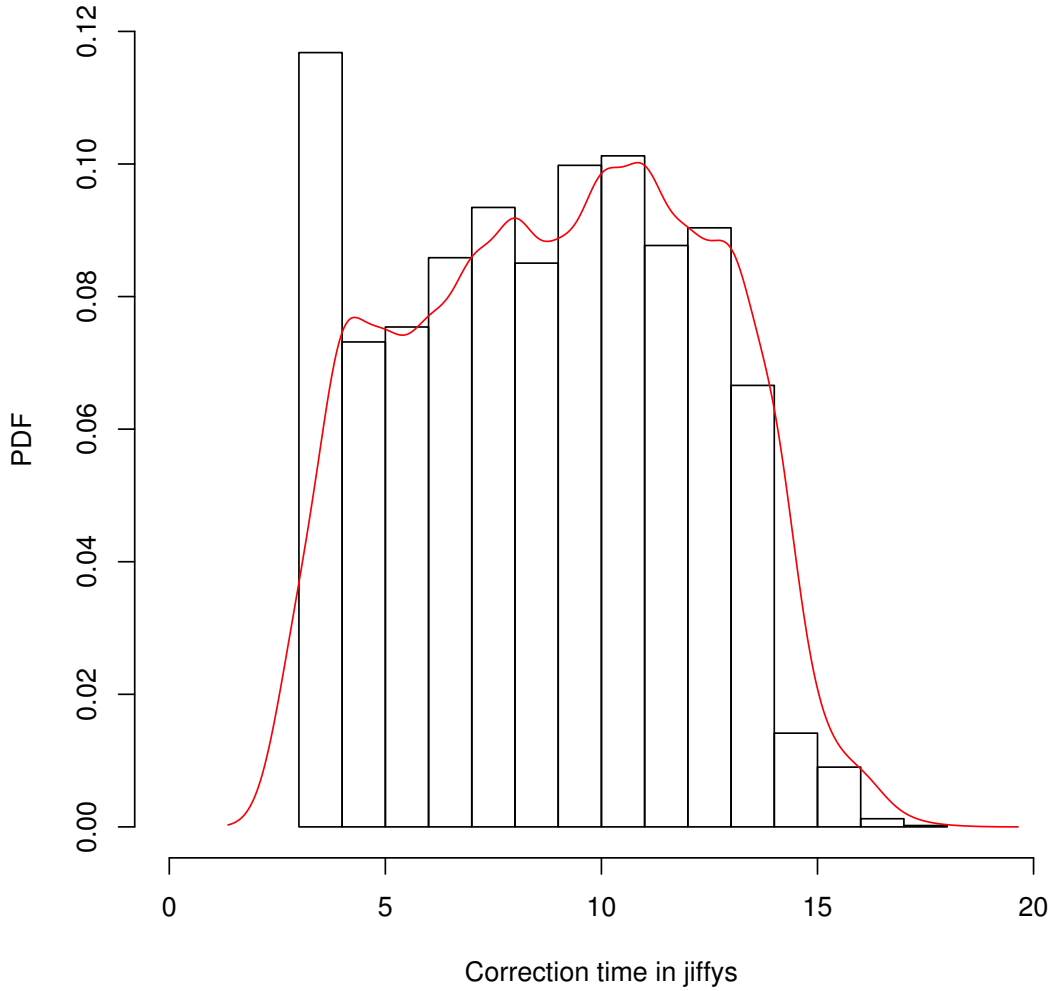


Figure 6. Histogram and P.D.F. of correction times.

Our first experiment measures the probability distribution of correction times (recall from Section 3 that it is $t2 - t1 + \delta$). As can be seen in Figure 6, correction time

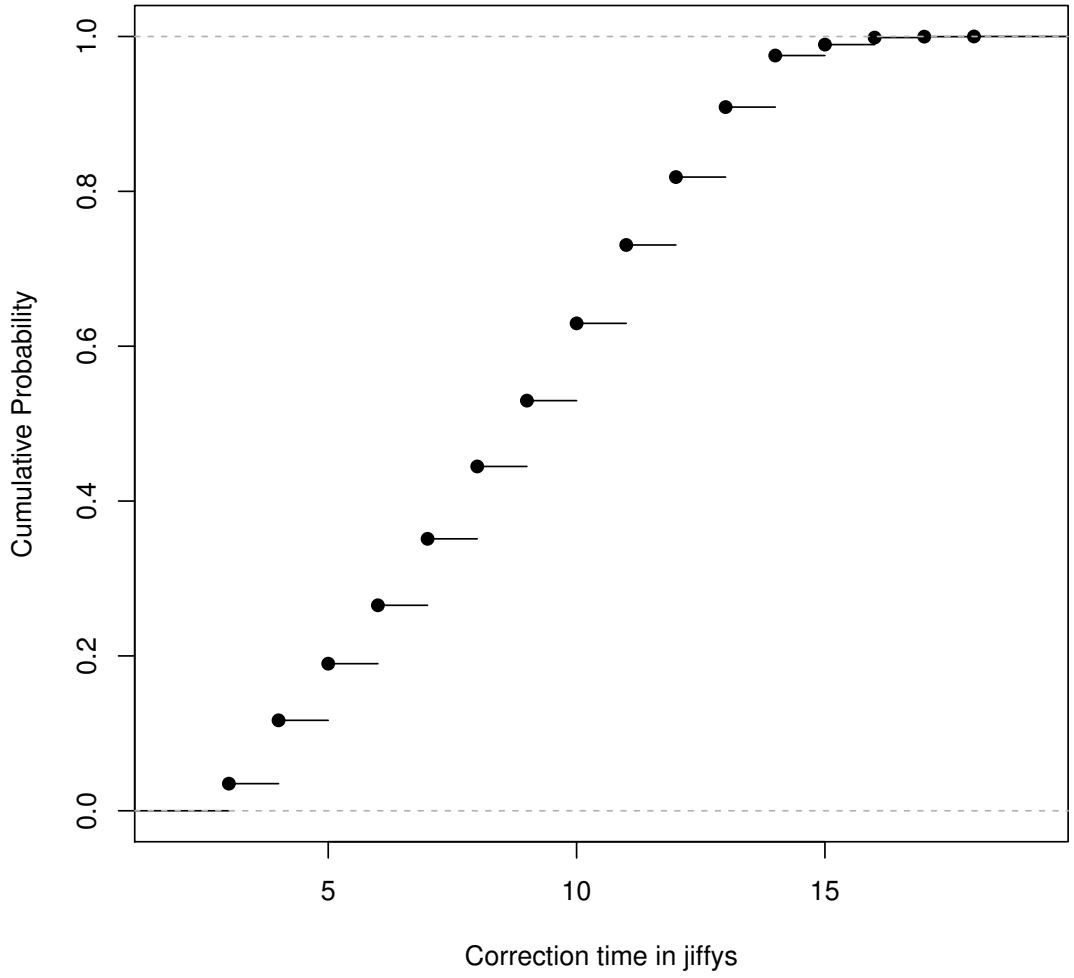


Figure 7. C.D.F. of correction times.

	mean (μs)	**std. deviation**
processing time	0.87 ± 0.0095	0.33
δ	0.88 ± 0.0093	0.33
correction time	9.01 ± 0.093	3.32
medium access	8.13 ± 0.093	3.31
$\mathbf{t4' - \bar{t4'}}$	-0.0047 ± 0.013	0.47

Table 1. Mean and standard deviation

varies mostly from 5 to 13 jiffys with an uniform distribution. We can see in Figure 7, which plots the cumulative density function of the correction times, that in about 80% of cases correction time is above 5 jiffys. This is a significant overhead and source of inaccuracy that FTSP+ is able to measure and compensate for. For our scenario the average

correction time delay is 9.01[1] jiffys as we can see in Table 1 where we have the mean with confidence interval of 95% and the standard deviation.

Delay in jiffys	0	1	2	3	4	5
Frequency	606	4271	0	1	1	1

Table 2. Delay frequency of receiver *processing time*.

Delay in jiffys	0	1	2	3	4	5
Frequency	581	4297	0	0	1	1

Table 3. Delay frequency for δ.

Our second experiment measures the probability distribution of δ and *processing time*. Recall from Section 3 that they are the radio interrupt processing time in the sender and the receiver, respectively. We can see in Table 2 and 3 that their values mostly range between 0 and 1 jiffy, with extremely rare cases that do not go beyond 5 jiffys in our experiments. Their averages are 0.87 and 0.88 jiffys, which can be considered equal. Therefore, those delays, that FSTP+ is not able to calculate and compensate for, are very small and do not compromise synchronization accuracy significantly.

We calculate an estimation of the error and compare our results with other protocols (some use MAC timestamp, some do not) and summarize the results in Table 4. Knowing MAC timestamping has an inherent 1 jiffy variation in accuracy and that FTSP has $1.5\mu s$ error per hop, we use a simple rule of three to estimate FTSP+ synchronization error per hop. FTSP and FSTP+ differ only for their timestamping technique, and as can be seen in Table 1 FTSP+ timestamping is on average 0.0047 jiffy higher than MAC timestamping ($t4' - \bar{t4'}$). Therefore, our rule of three is given in Equation 4.

$$\frac{1.5\mu s}{\text{FTSP+ sync error}} = \frac{1\textit{jiffys}}{(0.0047 + 1)\textit{jiffys}} \tag{4}$$

We also estimate what would be the error of FTSP without MAC timestamping (we call it mFTSP in the table), in which case timestamping is on average 9 jiffys higher than MAC timestamping ($medium\ access\ delay + processing\ delay = 8.13 + 0,87 = 9$). Equation 5 shows the rule of three for this estimation. As can be seen, RBS has a synchronization error about 10 times bigger than FTSP+.

$$\frac{1.5\mu s}{\text{mFTSP+ sync error}} = \frac{1\textit{jiffys}}{(9 + 1)\textit{jiffys}} \tag{5}$$

In TinyOS 1.x, where FTSP was first implemented and tested [Maróti et al. 2004], jiffys are in microsecond resolution by default. A jiffy in TinyOS 2.1.2 is approximately $1ms$, but there are compiler *TMilli* options to allow microsecond resolution. In future

[1]This value is scenario-dependent while it depends upon the scenario contention level.

[1]This value is scenario-dependent while it depends upon the scenario contention level.

[2]Values estimated based on average error in jiffys.

[3]Based in [Ranganathan and Nygard 2010].

MAC Timestamping		App Timestamping		
FTSP	**PulseSync**	**FTSP+**	**mFTSP**	**RBS**
$1.5\mu s$ [3]	$1.5\mu s$ [3]	$1.508\mu s$ [2]	$15\mu s$ [1 2]	$29\mu s$ [3]

Table 4. Average synchronization error per hop.

work, we want to use jiffys with microsecond resolution and measure the synchronization error per hop using a setup with GPIO pins for tracking events in an external and stable accuracy clock that record every exchanged messages.

4.1. Energy Consumption

The energy consumption was analyzed in experiments using the resources available on the Testbed [Lim et al. 2015]. Running the previous experiments we find the following result listed in Table 5.

FTSP	99.04 mJ
FTSP+	99.80 mJ

Table 5. Mean of consumption

The values represent the power consumption of CPU and radio communication of network (include the root and client node). We find close values in the tests, they are also related to synchronization frequency used on the network, for each synchronization message of FTSP the FTSP+ send another for correction, causing an extra consumption.

5. Conclusion

In this work, we presented a modified version of Flooding Time Synchronization Protocol to work without MAC layer timestamping. To keep accuracy as high as possible, we use radio interrupts to measure the instant nodes get medium access. Senders use correction messages in addition to synchronization messages in order to compensate their synchronization timestamps with medium access delay.

In our experiments, we ran tests on Micaz motes running TinyOS 2.1.2 to measure correction times, processing times and medium access time. We showed that medium access time is the main source of synchronization error and quantified it in a real testbed. We also showed that processing times are very small, on average 0.87 jiffys.

In future work, we want to precisely measure the synchronization error using a high accuracy external clock to measure timing error between synchronization events among the network motes. We also intend to consolidate the power consumption tests for different scenarios and see how much the variation in synchronization parameters affect in energy expenditure of WSN.

References

[Arampatzis et al. 2005] Arampatzis, T., Lygeros, J., and Manesis, S. (2005). A survey of applications of wireless sensors and wireless sensor networks. In *IEEE International Symposium on Intelligent Control, Mediterrean Conference on Control and Automation*, pages 719–724. IEEE.

[Baronti et al. 2007] Baronti, P., Pillai, P., Chook, V. W. C., Chessa, S., Gotta, A., and Hu, Y. F. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Comput. Commun.*, pages 1655–1695.

[Cardell-Oliver et al. 2004] Cardell-Oliver, R., Smettem, K., Kranz, M., and Mayer, K. (2004). Field testing a wireless sensor network for reactive environmental monitoring [soil moisture measurement]. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pages 7–12.

[Chong et al. 2014] Chong, F., Heck, M., Ranganathan, P., Saleh, A., and Wassel, H. (2014). Data center energy efficiency:improving energy efficiency in data centers beyond technology scaling. *Design Test, IEEE*, 31(1):93–104.

[Elson et al. 2002] Elson, J., Girod, L., and Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163.

[Elson and Estrin 2003] Elson, J. E. and Estrin, D. (2003). *Time synchronization in wireless sensor networks*. PhD thesis, University of California, Los Angeles.

[Ganeriwal et al. 2003] Ganeriwal, S., Kumar, R., and Srivastava, M. B. (2003). Timing-sync protocol for sensor networks. In *1st International Conference on Embedded Networked Sensor Systems*, pages 138–149. ACM.

[Gui and Mohapatra 2004] Gui, C. and Mohapatra, P. (2004). Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom '04, pages 129–143. ACM.

[Lenzen et al. 2009] Lenzen, C., Sommer, P., and Wattenhofer, R. (2009). Optimal clock synchronization in networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 225–238. ACM.

[Levis et al. 2005] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., et al. (2005). Tinyos: An operating system for sensor networks. In *Ambient intelligence*, pages 115–148. Springer.

[Lim et al. 2015] Lim, R., Maag, B., Dissler, B., Beutel, J., and Thiele, L. (2015). A testbed for fine-grained tracing of time sensitive behavior in wireless sensor networks. In *Local Computer Networks Workshops*.

[Maróti et al. 2004] Maróti, M., Kusy, B., Simon, G., and Lédeczi, Á. (2004). The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49. ACM.

[MEMSIC 2015] MEMSIC (2015). Micaz datasheet. `http://www.memsic.com`.

[Nazemi Gelyan et al. 2007] Nazemi Gelyan, S., Eghbali, A., Roustapoor, L., Yahyavi Firouz Abadi, S., and Dehghan, M. (2007). Sltp: Scalable lightweight time synchronization protocol for wireless sensor network. In *Mobile Ad-Hoc and Sensor Networks*, volume 4864, pages 536–547. Springer Berlin.

[Oliveira and Rodrigues 2011] Oliveira, L. M. and Rodrigues, J. J. (2011). Wireless sensor networks: a survey on environmental monitoring. *Journal of communications*, 6(2):143–151.

[Ranganathan and Nygard 2010] Ranganathan, P. and Nygard, K. (2010). Time synchronization in wireless sensor networks: a survey. *International journal of UbiComp (IJU)*, 1(2):92–102.

[Simon et al. 2004] Simon, G., Maróti, M., Lédeczi, Á., Balogh, G., Kusy, B., Nádas, A., Pap, G., Sallai, J., and Frampton, K. (2004). Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12. ACM.

[Sinopoli et al. 2003] Sinopoli, B., Sharp, C., Schenato, L., Schaffert, S., and Sastry, S. S. (2003). Distributed control applications within sensor networks. *Proceedings of the IEEE*, 91(8):1235–1246.

[Sommer and Wattenhofer 2009] Sommer, P. and Wattenhofer, R. (2009). Gradient clock synchronization in wireless sensor networks. In *International Conference on Information Processing in Sensor Networks*, pages 37–48.

[Sousa et al. 2014] Sousa, C., Carrano, R. C., Magalhaes, L., and Albuquerque, C. V. (2014). Stele: A simple technique for local delay estimation in wsn. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–6. IEEE.

[van Greunen and Rabaey 2003] van Greunen, J. and Rabaey, J. (2003). Lightweight time synchronization for sensor networks. In *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications*, pages 11–19. ACM.

[Wang and Ahn 2009] Wang, S. and Ahn, T. (2009). Mac layer timestamping approach for emerging wireless sensor platform and communication architecture. US Patent App. 12/213,286.

[Yang and Sikdar 2003] Yang, H. and Sikdar, B. (2003). A protocol for tracking mobile targets using sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pages 71–81. IEEE.

[Zanatta et al.] Zanatta, G., Bottari, G. D., Guerra, R., and Leite, J. C. B. Building a WSN infrastructure with COTS components for the thermal monitoring of datacenters. In *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea*.

Trilha Principal do SBRC 2016
Sessão Técnica 20
Segurança em Redes

Avaliação do Impacto da Segurança sobre a Fragmentação em Redes de Sensores Sem Fio na Internet das Coisas

Francisco Ferreira de Mendonça Júnior[1], **Obionor de Oliveira Nóbrega**[2], **Paulo Roberto Freire Cunha**[1]

[1] Centro de Informática - Universidade Federal de Pernambuco (UFPE)
CEP: 50740-540 – Recife – PE – Brasil

[2] Departamento de Estatística e Informática – Universidade Federal Rural de Pernambuco (UFRPE)
Recife – PE – Brasil

{ffmj, prfc}@cin.ufpe.br, obionor.nobrega@ufrpe.br

Abstract: *On the Internet of Things, devices with limitations in computational and network resources are connected directly to the Internet. In some scenarios, these devices need to exchange data whose length exceeds the amount of available space on its frames, causing fragmentation, which adversely impacts the performance and lifetime of these networks. This paper presents an analysis of the impact of fragmentation in the delivery delay of messages in a unicast transmission scenario in a 6LoWPAN network. The reduction in the transmission of one fragment of each device can cause reductions of 20% in the delay. The variation of the security level is considered feasible to reduce fragmentation.*

Resumo: *Na Internet das Coisas, dispositivos com limitações de recursos computacionais e de rede estão conectados diretamente à Internet. Em alguns cenários, essas redes precisam trafegar dados cujo comprimento excede a carga útil disponível nos quadros em sua camada de enlace. Isso causa fragmentação, que impacta negativamente em seu desempenho e tempo de vida. Este trabalho apresenta uma análise do impacto da fragmentação no atraso de entrega de mensagens num cenário de transmissão unicast numa rede 6LoWPAN. A manipulação do nível de segurança é utilizada para controlar a fragmentação. A redução de um fragmento na transmissão de cada dispositivo pode causar reduções de até 20% no atraso.*

1. Introdução

A Internet das Coisas tem surgido como um dos paradigmas que serve de base para a evolução da Internet do Futuro [Gubbi et al. 2013]. Trata-se da conexão de objetos à Internet através de sensores, atuadores e tecnologias sem fio. A adoção do IPv6 permite que cada objeto seja endereçado individualmente, tornando possível que eles se comuniquem e troquem dados mesmo sem a intervenção humana.

A conexão de objetos à Internet passa pela consolidação das Redes de Sensores e Atuadores sem Fio (RSASF) e das tecnologias de conexão dessas redes com a Internet. Geralmente a conexão com a Internet é realizada através de *Gateways*, dispositivos responsáveis por coletar e armazenar dados dos sensores de uma RSASF e transmiti-los quando necessário. [Rachedi et al. 2015] [Yick et al. 2008]

Nos últimos anos surgiram iniciativas que pretendem mudar o paradigma de conectividade das RSASF para que os dispositivos conectem-se à Internet sem intermediários. A subcamada 6LoWPAN (*IPv6 over Low power Wireless Personal Area Networks)* provê conectividade com a Internet para dispositivos que tenham camada de acesso ao meio e camada física diferenciadas. A adaptação consiste em tradução de endereços, compressão de cabeçalhos e tratamento de fragmentação [Montenegro et al. 2007]. A função do *gateway* foi substituída pela presença do *6LoWPAN Border Router* (6LBR), que já não armazena dados dos sensores, mas direciona as requisições e o envio de dados diretamente para os dispositivos. Dispositivos das RSASF passam a ser chamados de *6LoWPAN nodes* (6LN). [Shelby et al. 2012]

Quando conectadas à Internet, as redes 6LoWPAN entram em contato com redes com características diferenciadas, abrindo novos campos para o estudo de seu funcionamento. Uma das diferenças está relacionada à quantidade e ao tamanho dos dados trafegados. A unidade máxima de transmissão do IPv6 é de 1280 bytes [Deering and Hinden 1998]. Quando pacotes assim precisam trafegar por redes 6LoWPAN, eles passam por um processo de fragmentação.

Redes 6LoWPAN geralmente apresentam tráfego de dados, como sinalização e controle, que não excedem o espaço disponível em seus quadros [Kushalnagar et al. 2007]. Mas existem tipos de redes que precisam enviar registros de várias medições que não cabem em um único quadro. Esses registros podem variar de centenas de bytes a kilobytes . Alguns exemplos mostrados se referem ao monitoramento de ferrovias, cujos dados podem alcançam 7 KB; algumas aplicações de monitoramento de saúde podem alcanças 512 KB; e aplicações de monitoramento de vulcões podem alcançar 256 bytes. [Ludovici et al. 2014].

A fragmentação causa impactos negativos no desempenho, no consumo energético e pode causar perda de pacotes. Reduzir a fragmentação permite economizar os recursos escassos das redes 6LoWPAN [Hummen et al. 2013], [Silva et al. 2009], [Pope and Simon 2013], [Kuryla and Schönwälder 2011], [Ludovic, 2014], [Suh et al. 2008]. Este trabalho apresenta uma análise da variação do nível de segurança e sua relação com a fragmentação. Busca-se analisar em quais situações a variação do nível de segurança reduz a fragmentação. São propostos algoritmos que identificam essas situações utilizando apenas tecnologias de comunicação e segurança já padronizadas. Além disso, são analisados cenários onde a relação entre a quantidade de dispositivos, a quantidade de fragmentos e a modalidade de envio de mensagens constituem um ambiente mais amplo que os cenários comumente encontrados na literatura.

A próxima seção apresenta algumas características do padrão 802.15.4 com foco nos mecanismos de fragmentação e segurança. Na seção 3 serão discutidos trabalhos que tratam sobre fragmentação, suas falhas e as relações com o presente trabalho. Na seção 4 é feita uma análise da relação entre fragmentação e segurança. Busca-se investigar situações onde a redução do nível de segurança reduz a fragmentação, principalmente através da apresentação de equações que ajudam a identificar essas situações. Nas seções 5 e 6 se discute, através de um experimento e da apresentação dos resultados, a análise do impacto de fragmentação quando causada pela aplicação de segurança. Na seção 7 são apresentadas considerações finais e propostas para trabalhos futuros.

2. Padrão 802.15.4 – Fragmentação e Segurança

O padrão IEEE 802.15.4 especifica as camadas MAC e física para redes pessoais de baixa taxa de transferência (*Low Rate Wireless Personal Area Networks*, LR-WPAN) de. O foco da especificação foi manter simplicidade de implementação, baixo custo e baixo consumo energético [Daidone et al 2014]. Essas características permitiam uma ampla adoção do padrão pela comunidade de RSSF. [Sastry et al. 2004]

O padrão prevê 3 tipos de camada física que funcionam nas frequências 868 MHz (mega-hertz) , 915 MHz ou 2.4 GHz (giga-hertz). É possível alcançar velocidades de 20 kbps (kilobits por segundo), 40 kbps e 250 kbps, respectivamente, em cada frequência de operação. Independente da frequência de operação, a camada MAC utiliza o mesmo formato de quadro. Esse quadro possui, no máximo, 127 bytes [Callaway et al 2002]. As características diferenciadas de largura de banda e de formato de quadros fazem com que o padrão 802.15.4 necessite da subcamada 6loWPAN para se conectar à Internet.

2.1. Fragmentação em Redes 802.15.4

Quando um dado vindo da subcamada 6loWPAN não couber em apenas um quadro 802.15.4, ele deve ser fragmentado. Para isso é adicionado um cabeçalho auxiliar de fragmentação. Qualquer mecanismo de segurança, quando houver, é aplicado após o processo de fragmentação. [Raza et al. 2012][Montenegro et al. 2007]

O cabeçalho auxiliar de fragmentação varia de acordo com a quantidade de fragmentos. O primeiro fragmento recebe um cabeçalho auxiliar composto por 4 octetos. Esses octetos são divididos entre um preâmbulo de 5 bits, o campo "*datagram_size*" e o campo "*datagram_tag*". O cabeçalho auxiliar do primeiro fragmento pode ser visto na figura 1(a). A partir do segundo fragmento, o cabeçalho auxiliar ganha o campo "*datagram_offset*". O formato do cabeçalho auxiliar dos fragmentos subsequentes pode ser visto na figura 1(b). [Montenegro et al. 2007]

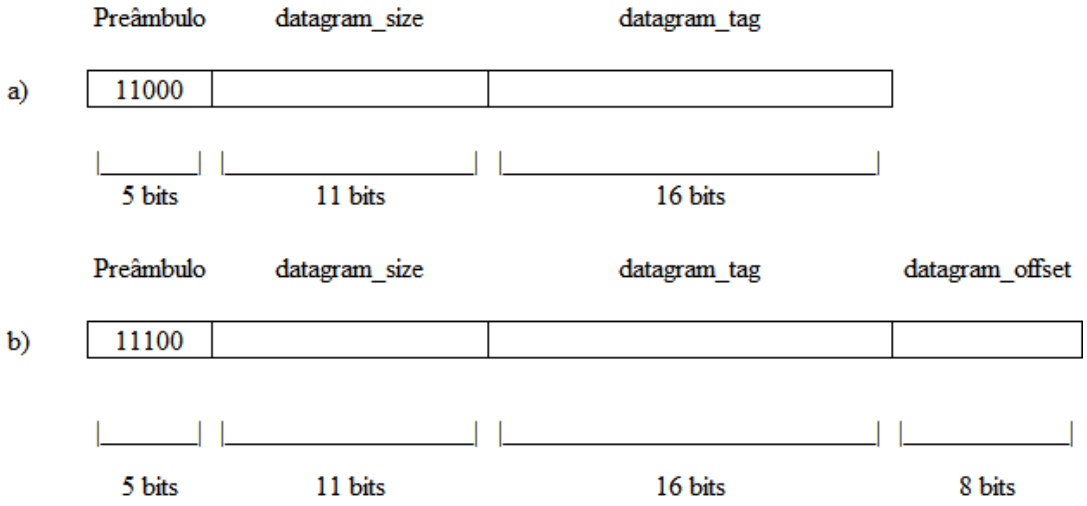


Figura 1: Formato dos cabeçalhos auxiliares de fragmentação para o primeiro fragmento (a) e para os demais fragmentos (b)

O preâmbulo identifica se aquele é o primeiro fragmento, chamado de FRAG1, ou algum fragmento subsequente, chamado de FRAGN. O campo "datagram_size" contém o comprimento do pacote completo, antes da fragmentação. Seu valor é comum

a todos os fragmentos, pois pode auxiliar na alocação de buffer caso os fragmentos cheguem fora de ordem. O campo "*datagram_tag*" contém um valor que identifica se um fragmento pertence a um determinado pacote. Esse valor é incrementado a cada novo pacote que necessita de fragmentação. O valor do campo "*datagram_offset*" indica o deslocamento de cada fragmento subsequente em relação ao primeiro fragmento. Ele permite que o pacote seja remontado de forma correta.

Redes 6LoWPAN são propensas a perdas e esses defeitos são agravados pela presença da fragmentação. O principal impacto para a transmissão de dados é que a fragmentação muda a modalidade de envio de mensagens. Geralmente, a taxa de geração de mensagens em uma rede é modelada por uma distribuição Poisson. Na presença de fragmentação, a rede se torna congestionada, uma vez que se considera que os fragmentos são transmitidos em rajadas [Pope and Simon 2013][Ludovic, 2014].

2.2. Segurança em Redes 802.15.4

Os mecanismos e opções de segurança no padrão 802.15.4 são conhecidos como a "subcamada de segurança" [Daidone et. al. 2014]. O padrão 802.15.4 prevê três modos de segurança: Modo Inseguro, Modo de ACL e o Modo Seguro. O Modo Inseguro não proporciona nenhuma proteção para os quadros transmitidos. No Modo de ACL (*Access Control List*), um dispositivo pode se comunicar apenas com dispositivos que estejam em uma lista pré-configurada. Quadros recebidos de outros dispositivos são descartados. Esse modo não fornece nenhuma garantia de confidencialidade, integridade ou proteção contra reenvio [Xiao et al. 2006].

O Modo Seguro prevê a utilização de 8 níveis de segurança para a camada de enlace e física. Os níveis podem ser vistos na Tabela 1. Esses níveis podem prover segurança nula, proteção de integridade, confidencialidade, ou proteção de integridade e confidencialidade combinadas. Existe também proteção contra reenvio de mensagens. [Xiao et al. 2006][Sastry et al. 2004]

Tabela 1. Níveis de segurança no padrão 802.15.4

Nível de Segurança	Descrição	Dados Criptografados	Proteção de Integridade e Autenticidade	Sobrecarga de segurança (bytes)
0	Unsecure			0
1	AES-CBC-MAC-32		X	4
2	AES-CBC-MAC-64		X	8
3	AES-CBC-MAC-128		X	16
4	AES-CTR	X		5
5	AES-CCM-32	X	X	9
6	AES-CCM-64	X	X	13
7	AES-CCM-128	X	X	21

O nível zero não prevê nenhuma proteção para as mensagens. A proteção de integridade é fornecida pela cifra AES-CBC-MAC-X. O X corresponde a extensão do Código de Integridade de Mensagem (*Message Integrity Code,* MIC). Essas mensagens não são criptografadas. Essa cifra protege a mensagem e seu cabeçalho com um código de integridade, que é calculado por blocos. O MIC é adicionado ao final da carga útil. [Sastry et al. 2004][Xiao et al. 2006]

A confidencialidade é provida pelas cifras AES-CTR e AES-CCM-X. A cifra AES-CTR protege os dados utilizando AES sobre blocos de dados. Para isso um dado é dividido em blocos de 16 Bytes. Cada bloco usa um contador diferente durante o processo de criptografia. O contador faz parte do Vetor de Inicialização (*Initialization Vector, IV*), também conhecido como *nonce*. O IV é formado pela conjunção de alguns campos como um marcador estático e o endereço do emissor. Também é formado por três contadores: o contador de quadros (4 Bytes), contador de chave (1 Byte) e o contador para os blocos (2 Bytes). O emissor inclui o contador de quadros e o contador de chave na carga útil do quadro [Sastry et al. 2004][Xiao et al. 2006]. Existem recomendações para a não utilização da cifra AES-CTR de forma isolada. A cifra não oferece proteção de integridade e de proteção contra reenvio de mensagens. [Sastry et al. 2004][Raza et al. 2012]

A cifra AES-CCM provê confidencialidade e proteção de integridade. Trata-se da aplicação dos dois mecanismos escritos anteriormente. Inicialmente é aplicado a proteção de integridade, que adiciona o MIC ao fim da carga útil. Em seguida, o processo de criptografia é aplicado sobre a carga útil e o MIC. O processo de criptografia adiciona os contadores à carga útil. Na figura 2 vemos o formato dos cabeçalhos e rodapés adicionais requeridos com a aplicação de segurança na camada física do padrão 802.15.4.

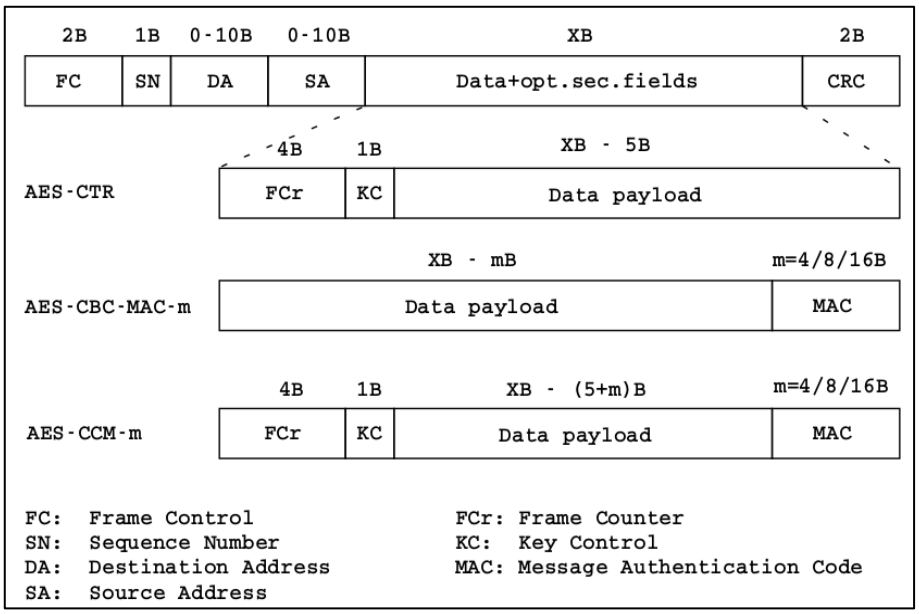


Figura 2: Formato do *quadro* do padrão 802.15.4 e sua relação com a segurança [Raza et al. 2012]

3. Trabalhos Relacionados

A preocupação com a fragmentação está presente desde o desenvolvimento das RSASF. Esse problema voltou à discussão quando se tornou necessário conectar esses dispositivos diretamente à Internet. Procura-se, desde então, propor mecanismos e melhorias nos mecanismos existentes para lidar com esse problema [Montenegro et al. 2007][Bormann and Shelby 2015]. Existem buscas de métodos para eliminar ou atenuar os efeitos da fragmentação, como pode ser visto em [Keoh et al. 2014]. Essa nova conectividade faz dispositivos das redes 6LoWPAN, cuja unidade máxima de

transmissão é 127 bytes, terem que lidar com o troca de dados diretamente com a Internet usando IPv6, cujo MTU (*Maximum Transmission Unity*) é de 1280 bytes [Hinden and Deering 1995].

Desde [Harvan and Schönwälder 2008] e [Cody-Kenny et al. 2008] até [Ludovici et al. 2014] essa preocupação com a fragmentação é constante. O trabalho de [Harvan and Schönwälder 2008] apresenta uma das primeiras análises do impacto da fragmentação em dispositivos limitados. Seus resultados apontam um crescimento no *round-trip time* usando ICMP (*Internet Control Message Protocol*) com mensagens *echo* que vão de 100 bytes a 1280 bytes. Apesar disso, seu cenário é simplificado com a presença de apenas um sensor. [Cody-Kenny et al. 2008] avaliaram o impacto no atraso e na perda de pacotes usando ICMP num *testbed* contendo 3 dispositivos e 1 estação base. Seus resultados mostraram aumento no atraso relacionado ao aumento no tamanho das mensagens. [Pope and Simon 2013] fizeram uma avaliação usando cenário com 16, 36 e 64 dispositivos, mas avaliaram apenas o processo de fragmentação até 2 fragmentos.

O estudo e correta avaliação do impacto na fragmentação não é necessário apenas em cenários de transmissão de dados. Alguns protocolos que auxiliem no gerenciamento, formação e manutenção de uma rede também podem precisar levá-la em consideração. [Kuryla and Schönwälder 2011] são levados a analisar o impacto da fragmentação no desenvolvimento de uma versão de SNMP para redes de dispositivos limitados. Algumas das mensagens trocadas pelo protocolo tinham tamanho próximo ou eram maiores que os 127 bytes permitidos no quadro do padrão 802.15.4. Essas mensagens sofriam fragmentação, e isso foi analisado para a avaliação do protocolo.

[Raza et al. 2013] preocupou-se com fragmentação no desenvolvimento de uma implementação de uma versão leve e segura do protocolo CoAP (*Constrained Application Protocol*). Métodos de compressão de mensagens de negociação do protocolo de segurança DTLS (*Datagram Transport Layer Protocol*) foram propostos. Mesmo assim, nem sempre era possível evitar fragmentação, pois algumas mensagens permaneciam com comprimento maior do que o máximo permitido.

O desenvolvimento de novos protocolos e a evolução dos antigos requer novas avaliações dos mecanismos de fragmentação existentes. O protocolo CoAP oferece a possibilidade de transferência de dados através de *blockwise transfer* [Bormann and Shelby 2015]. Trata-se de uma forma de dividir uma requisição ou uma resposta, de forma que cada parte seja transmitida como uma requisição independente nas camadas mais baixas da pilha de protocolos. Dessa forma, cada parte é transmitida e confirmada individualmente e pode ser retransmitida em caso de perda. Caso a requisição completa fosse enviada para as camadas inferiores e sofresse fragmentação, a perda de um dos fragmentos poderia gerar a retransmissão de toda a requisição. A retransmissão causa atrasos e aumento no consumo energético.

Pensando nisso, [Ludovic, 2014] realiza uma comparação entre fragmentação e *blockwise transfer*. Seus resultados apontam que a transmissão da requisição completa para que seja fragmentada na camada de rede pode ser mais eficiente que a utilização do mecanismo de *blockwise transfer*. Isso ocorre pois o CoAP demanda a troca de mais mensagens, gerando mais ocupação no canal que as confirmações da camada de rede.

[Rachedi et al. 2015] baseou-se no impacto da segurança para a construção de rotas pelo protocolo RPL em redes 6LoWPAN. Para isso ele se preocupou com o

impacto computacional de criptografar e decriptografar os dados. Esse impacto alimentava um controlador PID (*Proportional–Integral–Derivative controller*) para a construção de rotas do tipo AODV (*Ad-hoc On-demand Distance Vector*). Ainda assim não foram investigados os impactos que a fragmentação podia proporcionar, visto que a transmissão de dados acarreta a elevação do atraso e do consumo energético.

Apesar de estes trabalhos proporem melhorias, a maior parte dos cenários avaliados se constitui de cenários simplificados. As redes são formadas por poucos dispositivos. O modelo mais comum trata da avaliação utilizando apenas dois dispositivos, o que não condiz com a realidade de um cenário de Internet das Coisas. [Ludovic, 2014] realizou a avaliação em ambientes com mais dispositivos, mas não propôs mecanismos de eliminação ou atenuação dos efeitos da fragmentação.

4. Análise do Impacto da Fragmentação Causada pela Adição de Segurança

Visto que os cenários analisados na seção 3 são simplificados para o contexto da Internet das Coisas, é necessário avaliar a fragmentação em ambientes maiores e sob condições que simulem a troca de mensagens entre 6LN e a Internet. Técnicas que reduzam a fragmentação são necessárias, uma vez que seus impactos no aumento no atraso e taxa de entrega de mensagens são significativos [Harvan and Schönwälder 2008][Humenn, 2013][Pope and Simon 2013]. Serão analisadas as situações em que fragmentação pode acontecer devido à aplicação de segurança.

Esta análise é baseada na premissa de que a Qualidade de Serviço (*Quality of Service*, QoS) também é um fator importante para as aplicações de Internet das Coisas [Rachedi et al. 2015]. Estudos extensivos têm sido feitos em relação à ocupação de memória e consumo energético em RSASF [Yick et al. 2008]. Algumas métricas de QoS, como o atraso, tem sido deixadas em segundo plano em relação aos dois parâmetros citados.

É conhecido na literatura que o tamanho tem impactos no tempo de entrega das mensagens. Mesmo quando a fragmentação não é necessária, esse tempo pode variar dependendo da quantidade de mensagens, da quantidade de dispositivos na rede ou da quantidade de saltos. [Pope and Simon 2013][Rachedi et al. 2015]

Para esta análise serão utilizados apenas os níveis de segurança 5, 6 e 7, pois permitem o envio de dados criptografados e com proteção de integridade. Para que os dados sejam enviados com a máxima segurança habilitada é necessário acrescentar 21 bytes de informações de segurança em cada quadro, reduzindo a quantidade de dados que a aplicação pode enviar. Cada um dos níveis menores ocupa menos espaço por quadro.

Os limites da fragmentação correspondem à quantidade de bytes que podem ser economizados com a redução do nível de segurança. A quantidade de bytes necessários para a aplicação de cada nível está descrita na Tabela 1. À vista disso podem-se construir pseudocódigos para detectar os limites onde a redução do nível de segurança reduz o impacto da fragmentação.

No algoritmo 1, as variáveis "*SicsLowOneFragLen*", "*SicsLowFrag1Len*" e "*SicsLowFragNLen*" representam a quantidade de carga útil que um quadro pode transportar após a adição do MIC de acordo com o nível de segurança. A variável

"*SicsLowOneFragLen*" representa a carga útil de um quadro após a adição do MIC e quando não ocorre fragmentação. Essa variável representa uma quantidade maior de carga útil, pois nenhum cabeçalho de fragmentação foi adicionado.

Algoritmo 1: Algoritmo para identificação da quantidade de fragmentos de acordo com a nível de segurança

```
Algorithm Fragmentation Threshold Level
Input   len: Data Length
1 if len <= SicsLowOneFragLen then
2     fragments = 1
3 else
4     fragments = roundUp((len − SicsLowFrag1Len)/SicsLowFragNLen) + 1
5 return fragments
```

A variável "*SicsLowFrag1Len*" representa a carga útil do primeiro quadro de um dado fragmentado. Seu valor representa a quantidade de bytes disponíveis para dados após a adição do cabeçalho adicional de fragmentação e do MIC. Essa variável apresenta um valor maior que "*SicsLowFragNLen*". Isso acontece porque o cabeçalho de fragmentação nos fragmentos subsequentes é maior, reduzindo o espaço para carga útil nos fragmentos adicionais.

Os valores que podem ser atribuídos as variáveis do algoritmo 1 são relacionados com "XB", na figura 2. Esses valores são diferentes para cada nível de segurança e podem ser vistos na tabela 2. Na linha 1 do algoritmo 1, é feito o cálculo da quantidade de dados que não ativa a fragmentação. Este valor é representado pela variável "*SicsLowOneFragLen*". Valores maiores que o especificado na coluna "*SicsLowOneFragLen*" da tabela 2 provocam a fragmentação, de acordo com o nível de segurança.

Tabela 2. Valores para as variáveis do Algoritmo 1 de acordo com o nível de segurança

	SicsLowOneFragLen	**SicsLowFrag1Len**	**SicsLowFragNLen**
Level 5	91	87	86
Level 6	87	83	82
Level 7	79	75	74

A presença das "*SicsLowFrag1Len*" e "*SicsLowFragNLen*" na linha 3 do Algoritmo 1 é relacionada ao cabeçalho de fragmentação, que possui valores diferentes para o primeiro fragmento e para os fragmentos seguintes. No caso do nível 5, o valor 87 para "*SicsLowFrag1Len*" indica a quantidade de dados referente ao primeiro fragmento. A divisão por 86, referente à variável "*SicsLowFragNLen*", indica a quantidade de dados que os demais fragmentos podem transportar. Quanto maior o nível de segurança, menor é a quantidade de espaço disponível para dados. A mesma relação pode ser aplicada aos demais níveis de segurança.

O algoritmo 2 verifica se, para uma dada quantidade de dados, a redução no nível de segurança resultará na redução da quantidade de fragmentos. O acrônimo "FTL*" representa o "*Fragmentation Threshold Level* *", e faz referência ao algoritmo 1 quando este for alimentado com os valores da tabela 2, de acordo com o nível de segurança aplicado. No algoritmo 2, a quantidade de fragmentos resultante da aplicação de cada nível de segurança é comparada. Quando a quantidade de fragmentos for menor, para um nível menor de segurança, a redução no nível de segurança é acionada.

Algoritmo 2: Algoritmo para redução da quantidade de fragmentos

```
Algorithm Thresholds
Input    frag7: FTL7
         frag6: FTL6
         frag5: FTL5
1 if frag5 < frag6||frag6 < frag7||frag5 < frag7 then
2     decreaseSecurityLevel()
3 return fragments
```

Na figura 3 verifica-se o impacto do tamanho da mensagem na fragmentação quando a quantidade de dados que chega a subcamada 6LoWPAN está próxima dos limites de fragmentação. O gráfico foi aprimorado para demonstrar situações onde a redução no nível de segurança reduz a quantidade de fragmentos. Isso significa que esses pontos satisfazem as condições expostas nos algoritmos 1 e 2 para a redução da fragmentação. É possível observar que existe uma tendência a uma redução cada vez maior na quantidade de fragmentos conforme o tamanho na mensagem aumente Um mecanismo que faça utilização do nível de segurança para evitar fragmentação pode ser viável [Rachedi et al. 2015].

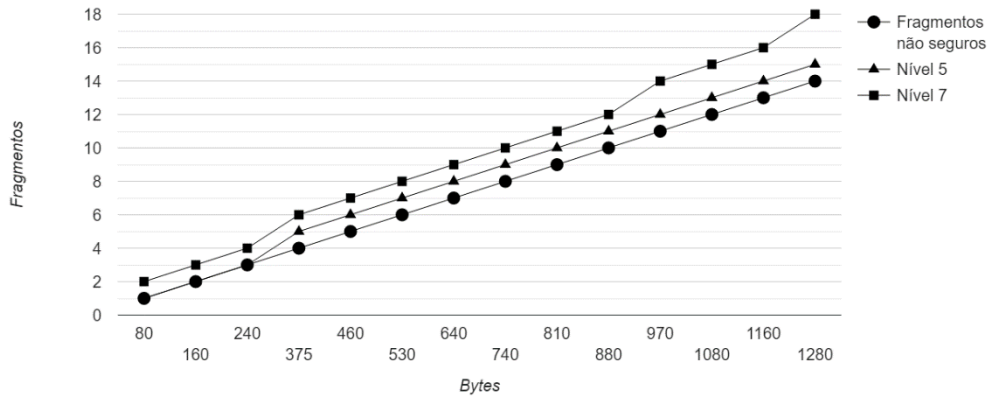


Figura 3: Aumento na quantidade de fragmentos relacionado a adição de segurança

5. Configuração do Experimento

Foram realizados experimentos para medir a economia de tempo ocasionada pela redução na quantidade de fragmentos. Os experimentos foram realizados usando ao simulador Cooja [Osterlind et al. 2006]. É possível simular dispositivos em três níveis: instruções em nível de máquina, nível de rede e sistema operacional. É possível coletar informações de cada um desses níveis.

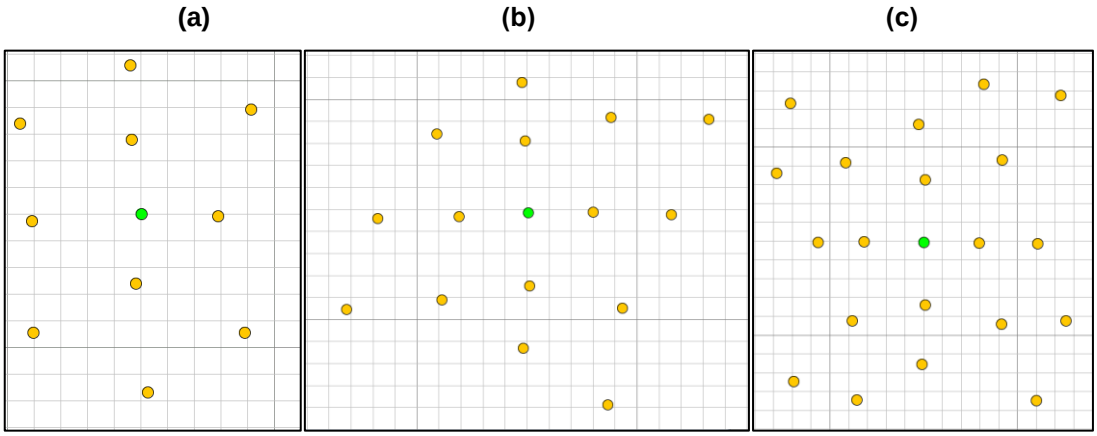
No simulador, foi configurada uma aplicação de entrega de mensagens utilizando UDP sobre 6loWPAN. Os dados são enviados a um receptor, no centro físico da rede em estrela, que emite uma mensagem na camada de aplicação quando recebe algum dado. Os experimentos foram feitos em redes de 11, 16 e 21 dispositivos [Ludovic, 2014]. Em cada rede, um dos dispositivos faz o papel de 6LBR e fica no centro da rede. Foram investigadas três taxas de envio de mensagens: 1, 5 e 10 mpm

(mensagens por minuto) [Ludovic, 2014]. O comprimento da mensagem foi controlado para simular o comportamento da manipulação de segurança sobre dados. Foi adotada a premissa de que o tempo gasto pelas operações criptográficas é desprezível quando realizado com o auxílio de aceleradores hardware [Lee et al. 2010].

Foram realizados experimentos em modelo fatorial completo para cada rede (3), cada quantidade de fragmentos (6) e cada modalidade de envio de mensagens (3), totalizando 54 tipos para simulação. Cada uma dessas simulações foi repetida por 10 vezes. Cada experimento foi executado durante 1 hora em tempo de simulação, de forma que são geradas até 13000 mensagens. O simulador gera sementes aleatórias da ordem de 19 dígitos para cada replicação. Cada experimento foi executado durante 1 hora em tempo de simulação, de forma que são geradas até 13000 mensagens.

A partir de cada uma das 10 repetições é gerada uma média. As amostras de 10 médias provenientes de simulações de quantidades de fragmentos adjacentes são comparadas. Para a comparação, são usados testes estatísticos para a diferença de duas amostras com intervalo de confiança de 90%. A hipótese alternativa do teste é que o atraso medido na rede que transmite mais fragmentos é maior que o atraso medido na rede com quantidade de fragmentos subjacente. Na figura 4 vemos os cenários da simulação para 11 dispositivos (a), 16 dispositivos (b) e 21 dispositivos (c).

Figura 4: Cenário da simulação



6. Análise dos Resultados

Pode-se observar, nos gráficos das figuras 5, 6 e 7, redução da fragmentação apresenta redução no atraso. Esses valores representam a redução no atraso médio quando o dado é enviado de um 6LN para o 6LBR. Porém, é possível observar que o aumento da quantidade de dispositivos e o aumento da quantidade de mensagens podem tornar os ganhos menos relevantes, uma vez que a rede está congestionada o tempo todo.

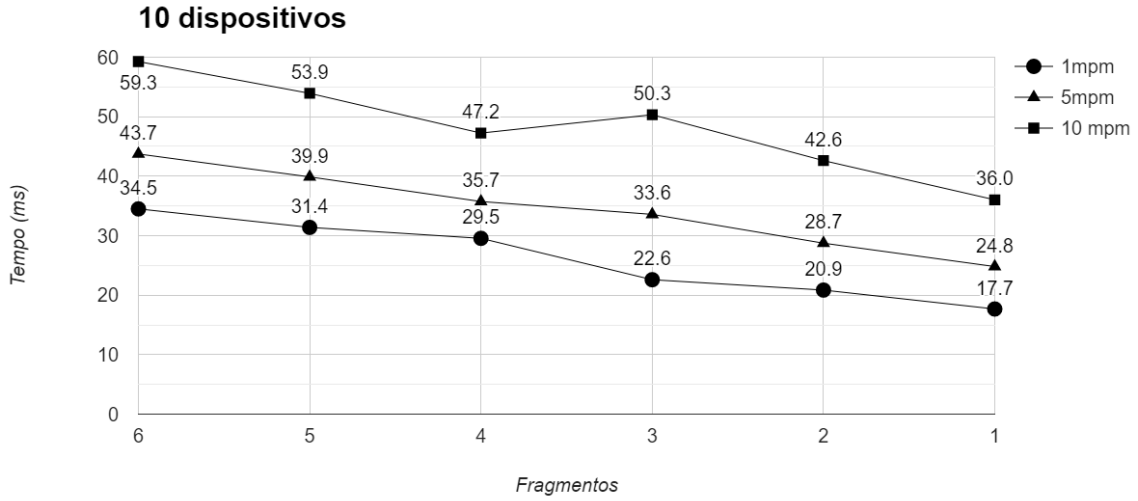


Figura 5: Redução média no atraso em redes de 10 dispositivos sob diferentes taxas de geração de dados

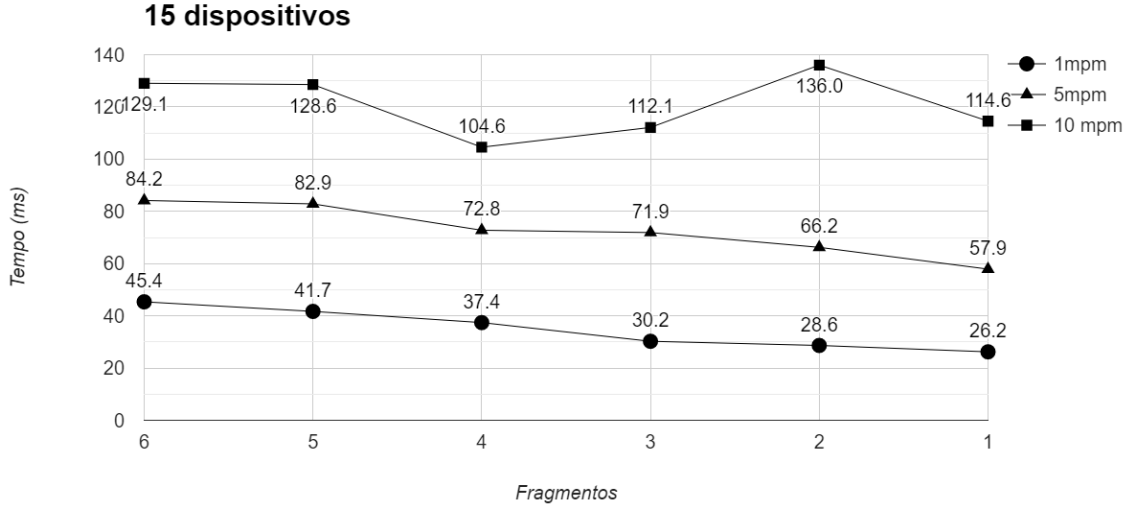


Figura 6: Redução média no atraso em redes de 15 dispositivos sob diferentes taxas de geração de dados

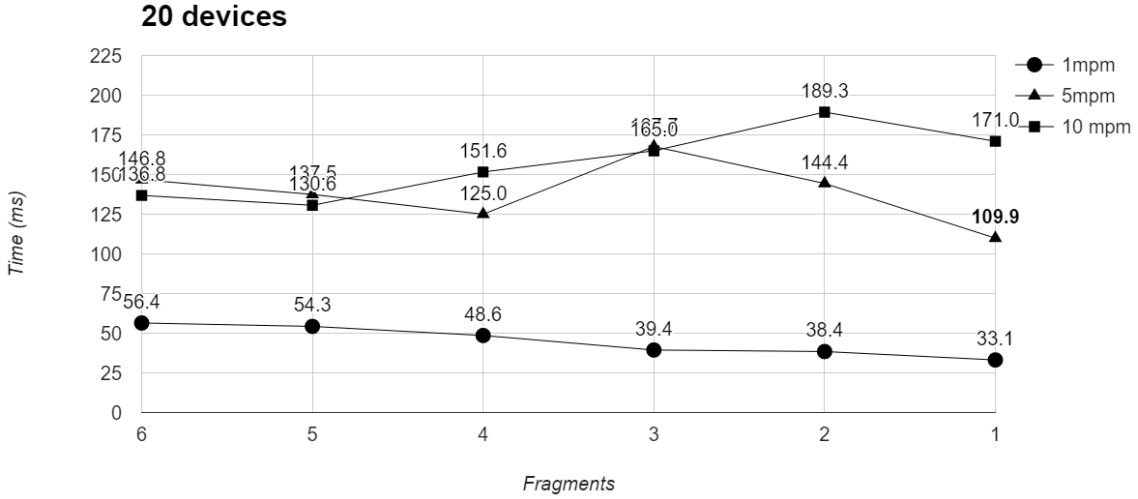


Figura 7: Redução média no atraso em redes de 10 dispositivos sob diferentes taxas de geração de dados

Na figura 5 é possível observar a redução no atraso médio da rede quando ocorre a redução de segurança na rede de 10 dispositivos. Nas modalidades de 1 mpm e 5 mpm todas as reduções apresentam ganhos significativos. Nas figuras 5, 6 e 7, no envio de 10 mpm, não existem ganhos estatisticamente significativos, apesar de existirem ganhos percentuais. Esse comportamento ocorre devido ao congestionamento da rede [Pope and Simon 2013] [Ludovic, 2014].

Redes 6LoWPAN cuja frequência de geração de dados está entre 1 e 5 mpm apresentam decréscimo significativo no atraso quando ocorre redução da fragmentação. Redes com até 10 dispositivos apresentam ganhos em todas as taxas.

Em redes com 15 e 20 dispositivos, os ganhos são expressivos quando o tráfego da rede é baixo. Nas redes com 15 dispositivos, os ganhos existem quando as mensagens são enviadas a, no máximo, 5 mpm. Até esse limite, são encontrados ganhos próximos a 20%. Nas redes de 20 dispositivos, os ganhos se concentram quando as mensagens são enviadas a 1 mpm. Quando a quantidade de mensagens aumenta, os ganhos passam a ser menos perceptíveis. Mesmo assim ainda são encontrados ganhos próximos a 20% nessas redes.

7. Considerações Finais e Trabalhos Futuros

Este trabalho avaliou as situações em que a redução do nível de segurança do padrão 802.15.4 auxilia na redução dos efeitos da fragmentação no atraso de redes 6LoWPAN. Foram propostos algoritmos para identificar essas situações e acionar a redução no nível de segurança. Foi possível analisar que existem casos em que ocorrem reduções de até 20% no atraso. As reduções ocorrem com mais significância em redes de 10 dispositivos. Em redes de 15 e 20 dispositivos os ganhos são significativos quando a taxa de transmissão de mensagens não ultrapassa 5 mpm. Conclui-se que a redução do nível de segurança é método viável para reduzir fragmentação e reduzir o atraso em redes 6LoWPAN.

Trabalhos futuros podem propor outras técnicas que utilizem a flexibilidade existente na subcamada de segurança do padrão 802.15.4, como demonstrado na seção 2.2, para economizar energia e diminuir o atraso no envio das mensagens. Podem ser

investigadas mais situações onde ganhos são expressivos. Também podem ser estudados os casos em que a criptografia é realizada por software, uma vez que parte dos rádios ainda não fornece suporte completo à criptografia realizada por hardware.

Referências

Bormann, C., & Shelby, Z. (2015). Blockwise Transfers in CoAP draft-ietf-core-block-18. Available online: http://tools. ietf. org/html/draft-ietf-core-block-18 (accessed on 25 nov 2015).

Cody-Kenny, B., Guerin, D., Ennis, D., Simon Carbajo, R., Huggard, M., & Mc Goldrick, C. (2009, October). Performance evaluation of the 6LoWPAN protocol on MICAz and TelosB motes. In Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks (pp. 25-30). ACM.

Callaway, E., Gorday, P., Hester, L., Gutierrez, J. A., Naeve, M., Heile, B., & Bahl, V. (2002). Home networking with IEEE 802. 15. 4: a developing standard for low-rate wireless personal area networks. IEEE Communications magazine, 40(8), 70-77.

Daidone, R., Dini, G., Anastasi, G. (2014). On evaluating the performance impact of the IEEE 802.15. 4 security sub-layer. Computer Communications, 47, 65-76.

Deering, S. E. Hinden, R (1998). Internet protocol, version 6 (IPv6) specification. (No. RFC 2460).

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.

Harvan, M., & Schönwälder, J. (2008). TinyOS Motes on the Internet: IPv6 over 802.15. 4 (6LoWPAN). PIK-Praxis der Informationsverarbeitung und Kommunikation, 31(4), 244-251.

Hinden, R., & Deering, S. (1995). Internet protocol, version 6 (IPv6) specification.

Hummen, R., Hiller, J., Wirtz, H., Henze, M., Shafagh, H., & Wehrle, K. (2013, April). "6LoWPAN fragmentation attacks and mitigation mechanisms". In Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks (pp. 55-66). ACM.

Keoh, S. L., Kumar, S. S., & Tschofenig, H. (2014). Securing the internet of things: A standardization perspective. Internet of Things Journal, IEEE, 1(3), 265-275.

Kuryla, S., & Schönwälder, J. (2011). Evaluation of the resource requirements of SNMP agents on constrained devices. In Managing the Dynamics of Networks and Services (pp. 100-111). Springer Berlin Heidelberg.

Kushalnagar, N., Montenegro, G., & Schumacher, C. (2007). IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals (No. RFC 4919).

Lee, J., Kapitanova, K., & Son, S. H. (2010). The price of security in wireless sensor networks. Computer Networks, 54(17), 2967-2978.

Ludovici, A., Marco, P. D., Calveras, A., & Johansson, K. H. (2014). Analytical model

of large data transactions in CoAP networks. Sensors, 14(8), 15610-15638.

Montenegro, G., Kushalnagar, N., Hui, J., & Culler, D. (2007). Transmission of IPv6 packets over IEEE 802.15. 4 networks (No. RFC 4944).

Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., & Voigt, T. (2006, November). Cross-level sensor network simulation with Cooja. In Local Computer Networks, Proceedings 2006 31st IEEE Conference on (pp. 641-648). IEEE.

Pope, J., & Simon, R. (2013). The Impact of Packet Fragmentation and Reassembly in Resource Constrained Wireless Networks. CIT. Journal of Computing and Information Technology, 21(2), 97-107.

Rachedi, A., & Hasnaoui, A. (2015). Advanced quality of services with security integration in wireless sensor networks. Wireless Communications and Mobile Computing, 15(6), 1106-1116.

Raza, S., Duquennoy, S., Höglund, J., Roedig, U., & Voigt, T. (2012). Secure communication for the Internet of Things—a comparison of link layer security and IPsec for 6LoWPAN. Security and Communication Networks.

Raza, S., Shafagh, H., Hewage, K., Hummen, R., & Voigt, T. (2013). Lithe: Lightweight secure CoAP for the internet of things. Sensors Journal, IEEE, 13(10), 3711-3720.

Sastry, N., & Wagner, D. (2004, October). Security considerations for IEEE 802.15. 4 networks. In Proceedings of the 3rd ACM workshop on Wireless security (pp. 32-42). ACM.

Shelby, Z., Chakrabarti, S., Nordmark, E., & Bormann, C. (2012). Neighbor discovery optimization for IPv6 over low-power wireless personal area networks (6LoWPANs) (No. RFC 6775).

Silva, R., Silva, J. S., & Boavida, F. (2009). Evaluating 6LoWPAN implementations in WSNs. Proceedings of 9th Conferncia sobre Redes de Computadores Oeiras, Portugal, 21.

Suh, C., Mir, Z. H., & Ko, Y. B. (2008). Design and implementation of enhanced IEEE 802.15. 4 for supporting multimedia service in Wireless Sensor Networks. Computer Networks, 52(13), 2568-2581.

Xiao, Y., Chen, H. H., Sun, B., Wang, R., & Sethi, S. (2006). MAC security and security overhead analysis in the IEEE 802.15. 4 wireless sensor networks. EURASIP Journal on Wireless Communications and Networking, 2006(2), 81-81.

Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless sensor network survey. Computer networks, 52(12), 2292-2330.

Um Controle de Autenticação de Handover Vertical para Prevenir Ataque de Repetição em Redes Heterogêneas Sem Fio

Adi Marcondes[1], **Michele Nogueira**[1], **Aldri Santos**[1]

[1]Núcleo de Redes Sem-Fio e Redes Avançadas (NR2) – UFPR
Caixa Postal 19.081 – 81.531.980 – Curitiba – PR – Brasil

{an.marcondes, michele,aldri}@inf.ufpr.br

Abstract. *The wireless heterogeneous networks allows increased compliance data transmission, as they consider interoperability in differents networks technologies. Moreover, the authentication service supports interoperability and admits access control in the devices. In the literature, several authentication schemes have been proposed for security this service during the transition in networks, but these schemes do not prevent effectively replay attack. Thus, this paper proposes the authentication scheme CAH to prevent replay attack in heterogeneous networks. The scheme consists of a unified authentication between different network technologies using the matching method and the uniqueness of message verification in transition networks. The mechanism checks the time and amount of messages, and sends information of the mobile devices to other networks via the overlay. The evaluation of this scheme compares the results with the UHA scheme using metrics of safety and performance.*

Resumo. *As redes heterogêneas sem fio possibilitam uma maior abrangência à transmissão de dados, visto que elas consideram a interoperabilidade em redes de tecnologias diferentes. Para isso, o serviço de autenticação oferece suporte à interoperabilidade e permite o controle de acesso dos dispositivos nas redes. Na literatura, vários esquemas de autenticação foram propostos para a segurança deste serviço durante a transição nas redes, porém estes esquemas não previnem o ataque de repetição de maneira eficaz. Diante disso, este trabalho propõe o esquema de autenticação CAH para prevenir o ataque de repetição em redes heterogêneas. O esquema consiste de uma autenticação unificada entre redes de tecnologias diferentes utilizando o método de pareamento e a verificação da unicidade das mensagens na transição de redes. O mecanismo verifica o tempo e a quantidade de mensagens, e envia informações dos dispositivos móveis para outras redes através da sobreposição. A avaliação deste esquema compara os resultados com o esquema UHA usando métricas de segurança e desempenho.*

1. Introdução

As redes heterogêneas sem fio (HetNets) abrangem várias tecnologias de redes e permitem que os usuários portadores de dispositivos computacionais móveis trafeguem por diferentes áreas de transmissão [Damnjanovic et al. 2011]. Deste modo, a proliferação destes dispositivos resultou no aumento intensivo do fluxo de dados que possui o suporte da interoperabilidade nas redes [Ranjan et al. 2014]. A interoperabilidade representa um requisito das HetNets para conectividade e sustentação dos serviços usados pelos usuários.

O *handover* vertical consiste de um processo da rede para a interoperabilidade que efetua a transferência da conexão [Xenakis et al. 2014]. Contudo, a execução do processo de *handover* vertical nas redes necessita de um serviço essencial chamado de autenticação.

A autenticação baseia-se em um serviço que auxilia os sistemas computacionais no reconhecimento dos usuários [Kent et al. 2015]. Este serviço lida com a segurança e o controle de acesso dos dispositivos móveis nas redes [Sandhu and Samarati 1994]. Apesar disso, o serviço de autenticação está sujeito a problemas de desempenho e segurança quando existe a necessidade do dispositivo móvel efetuar o *handover* vertical [He et al. 2015]. Os problemas do serviço de autenticação acarretam no aumento do custo de processamento dos dispositivos móveis e na descontinuidade ou atraso dos serviços usados pelos usuários das redes sem fio. Além dos prejuízos no desempenho, a ausência de confidencialidade na comunicação dos dispositivos móveis e da rede resultam em ataques que visam o acesso não autorizado e a perda de conexão afetando a disponibilidade dos serviços. Alguns ataques de autenticação tem como característica a obtenção dos quadros de transmissão durante a associação dos dispositivos móveis nas redes heterogêneas. Um dos principais ataques, chamado de ataque de repetição, representa a captura e replicação dos quadros de transmissão na comunicação das entidades das redes. Este ataque, nas redes heterogêneas, tem como meta o acesso não autorizado em um ponto de acesso através os quadros de autenticação dos dispositivos móveis legítimos obtidos durante o *handover* vertical.

Várias abordagens na literatura buscam resolver os problemas no serviço de autenticação que envolvem a segurança e o desempenho. Algumas soluções de autenticação têm como base a identidade dos dispositivos móveis [Cao et al. 2012, Feng and Jiao 2012, Li et al. 2012]. Outras soluções utilizam a distribuição da autenticação [Cheng et al. 2014] e formação de grupos de entidades [Fu et al. 2013] para proteger a rede contra o acesso não autorizado sem afetar o desempenho dos serviços de rede. No entanto, estas estratégias não garantem a segurança do serviço de autenticação no contexto de HetNets em razão de não considerarem as diferentes características de transmissão de dados [Cao et al. 2014]. Com intuito de melhorar o desempenho, algumas abordagens utilizam a criptografia baseada em pareamento na geração de identidades [He et al. 2012] para otimizar o tempo de execução da autenticação no serviço de *handover*. Na tentativa de melhorar o desempenho do processo de autenticação, estas soluções apresentam vulnerabilidades na transição entre as redes heterogêneas.

Os métodos de segurança utilizados na proteção da autenticação fundamentam-se no tempo de transmissão dos quadros e requisição de conexão entre o dispositivo móvel e o ponto de acesso da rede [Yu et al. 2013]. No entanto, o tempo de transmissão dos quadros assume apenas redes homogêneas, ou seja, com características de transmissão semelhantes [Hu et al. 2003]. Logo, a integração dos esquemas de autenticação nas HetNets oferece uma solução mais robusta contra o ataque de repetição. Ademais, métodos como a validação dos quadros de transmissão recebidos pelas redes e a distribuição de autenticação dos dispositivos entre as entidades autenticadoras anulam o acesso indevido.

Este trabalho apresenta o controle de autenticação de *handover* (CAH) que possui o objetivo de prevenir o ataque de repetição na autenticação do serviço de *handover* vertical dos dispositivos móveis nas redes heterogêneas. O CAH compreende duas fases de autenticação que envolvem os dispositivos móveis e as redes heterogêneas, e um me-

canismo de proteção que analisa os quadros de transmissão dos dispositivos móveis. Para avaliar o controle de autenticação CAH foram utilizadas métricas de segurança e de desempenho. Além disso, os resultados foram comparados com o esquema de autenticação UHA. As simulações mostram que o CAH dispõem de uma segurança mais eficaz contra o ataque de repetição nas HetNets e um melhor desempenho do serviço de autenticação no *handover* vertical.

O artigo está organizado desta forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha o funcionamento do controle de autenticação proposto. A Seção 4 mostra a avaliação de segurança e desempenho do esquema. Finalmente, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Os esquemas de autenticação com base na identidade utilizam as informações de identificação dos dispositivos móveis para autorizar o acesso à rede. O esquema UHA [Cao et al. 2012] sucede inicialmente da autenticação do dispositivo móvel através da rede de maior sobreposição de transmissão sem fio, onde existe uma entidade que cria as identidades e chaves dos dispositivos móveis. Apesar disso, os quadros de transmissão continuam vulneráveis e ocorrem atrasos no processo de autenticação do serviço de *handover* nas redes. O trabalho WAPI [Feng and Jiao 2012] se baseia na identidade e usa o protocolo de combinação lógica (PCL) [Cremers 2008]. Entretanto, este trabalho não oferece segurança contra ataques de quadros de transmissão na comunicação das entidades das redes. O esquema de autenticação LRA [Li et al. 2012] propôs uma autenticação para o serviço de *handover* com a pretensão de reduzir o processamento computacional e prover o anonimato para os dispositivos móveis sem a necessidade de um servidor de autenticação residencial. A prevenção dos ataques que usam os quadros de transmissão falha em razão da metodologia de validação da transmissão permitir quadros repetidos.

O trabalho DDA [Cheng et al. 2014] consiste de uma abordagem de autenticação distribuída. Este controle de autenticação tem como prioridade o anonimato e a distribuição da autenticação para solucionar os problemas de segurança e desempenho nas redes. Através do reconhecimento de dispositivos móveis e pontos de acessos como membros de um grupo, os algoritmos de assinatura adotados nesta abordagem fornecem a verificação de identidade. Os dispositivos móveis assinam as mensagens em nome deste grupo para se autenticar nas redes. Isto oferece uma resistência aos ataques que visam o anonimato e a ausência da disponibilidade. Esta estratégia não inviabiliza o ataque de repetição quando existe uma rede maliciosa que obtém informações de um dispositivo móvel e assim utilizadas para acesso em outra rede.

O controle de autenticação GHAP [Fu et al. 2013] estabelece um esquema baseado em grupos de *handover* para os dispositivos móveis. Este controle de autenticação consiste de três fases para desenvolvimento, inicialização e autenticação no handover. Além disso, o GHAP cria um contexto de segurança para os membros do grupo de *handover* para o ponto de acesso da rede destino usando o método SCT (*Security Context Transfer*). Isto acontece durante a fase de autenticação do dispositivo móvel. Este esquema tem o objetivo de resistir contra o ataque dominó existente nos métodos SCT. No entanto, os ataques que visam os quadros de comunicação tem efeitos sobre a autenticação.

O esquema PairHand [He et al. 2012] usa a criptografia baseada em pareamento

para a geração de identidades e chaves. Ademais, o objetivo deste esquema tem o propósito a alta eficiência no desempenho dos serviços da rede. Para isso, o esquema adota pseudônimos para proteger a privacidade e requerem maior capacidade de armazenamento dos dispositivos móveis. Isto possibilita o pré-processamento de um grande conjunto de pseudônimos de autenticação no servidor de gerência central. Apesar disso, a abordagem não considera a sobreposição nas redes heterogêneas para o controle de acesso. Isto permite que ataques de quadros sejam executados.

3. Controle de autenticação

Esta seção apresenta o controle de autenticação proposto CAH para prevenir o ataque de repetição na autenticação durante o processo de *handover* nas redes heterogêneas. Primeiramente apresenta-se o modelo da rede assumido para o controle de autenticação e o modelo de ataque. Em seguida, a especificação do esquema de autenticação que consiste das fases de inicialização e de autenticação, e o mecanismo de proteção são abordados.

3.1. Modelo da rede

As redes heterogêneas agregam diversos dispositivos computacionais que possuem características e funcionalidades diferentes [Heath et al. 2013].No modelo proposto, estes dispositivos computacionais consistem dos nós ou dispositivos móveis dos usuários representados pelo conjunto *N*, dos pontos de acesso (*AP*) e dos autenticadores (*A*). Cada autenticador compreende uma rede que se encontra no conjunto de redes heterogêneas.

A HetNet é composta pelo conjunto de n nós, móveis ou fixos, denotado por *N* = $\{n_1, n_2, n_3, ..., n_i\}$ tal que $n_i \in N$. Estes nós são classificados em nós comuns (*Nc*) e nós atacantes (*Na*), onde *Nc* $\subseteq$ *N*, *Na* $\subseteq$ *N* e *Nc* $\bigcup$ *Na* = *N*. Além disso esses nós possuem relação com os pontos de acesso das redes heterogêneas. Os pontos de acesso são representados por *AP*= $\{ap_1, ap_2, ap_3, ..., ap_i\}$ tal que $ap_i \in AP$, sendo que, o *ap* representa um ponto acesso específico dentro do conjunto de ponto de acesso *AP* em uma rede. Cada rede possui um autenticador (*A*) que realiza a autenticação dos nós. Logo o conjunto de redes heterogêneas (*H*) = $\{A_1, A_2, A_3, ..., A_i\}$.

Um ou mais elementos do conjunto de pontos de acesso *AP* está associado com um elemento do conjunto de autenticadores *A* de tal forma que são denotados por $F : AP \rightarrow A$. Os elementos do conjunto de nós *N* tem relação com elementos dos conjuntos *AP* e *A*. Logo, a relação pode ser representada por $N^{ap,a}$. A comunicação entre os dispositivos móveis e as redes acontece no meio sem fio. A comunicação nas redes heterogêneas ocorre através de um canal seguro, e isto impossibilita ataques na comunicação das entidades das redes (*H*).

3.2. Modelo do ataque

As HetNets estão sujeitas a diversos tipos de ataques que utilizam os quadros de autenticação. Neste trabalho é considerado o ataque de repetição. Neste ataque, os atacantes realizam a captura dos quadros de transmissão da autenticação enviados de um dispositivo móvel para um ponto de acesso. Os quadros capturados podem ser decifrados pelos atacantes e replicados no ponto de acesso. A replicação dos quadros tem como objetivo o acesso não autorizado aos recursos da rede. Para isso, os quadros de replicação devem ser validados por uma entidade autenticadora de uma rede presento no conjunto de redes heterogêneas (*H*).

O dispositivo móvel (*Nc*) efetua uma requisição para se conectar a uma rede com o ponto de acesso (*AP*). Logo, esta requisição no meio sem fio é enviada em formato de quadros de transmissão no qual, o *AP* reconhece a mensagem e envia para o autenticador *A* da rede. Um dispostivo móvel malicioso *Na* executa a captura dos quadros de transmissão da autenticação de *Nc*. Através da captura, o atacante pode replicar os quadros para efetuar uma inconsistência de informações no autenticador *A* e assim obter acesso não autorizado.

O ataque de repetição pode ser executado para atacar o dispositivo móvel. Neste caso, o *Na* replica os quadros de transmissão enviados pelo *AP* para o *Nc* a fim de confundir o dispositivo. Com esta forma de ataque, o atacante simula um *AP* falso para diversas finalidades como roubo de informações e também realizar outros ataques maliciosos. O ataque visando o usuário móvel também pode ser replicado de forma constante para esgotar os recursos limitados do *Nc*.

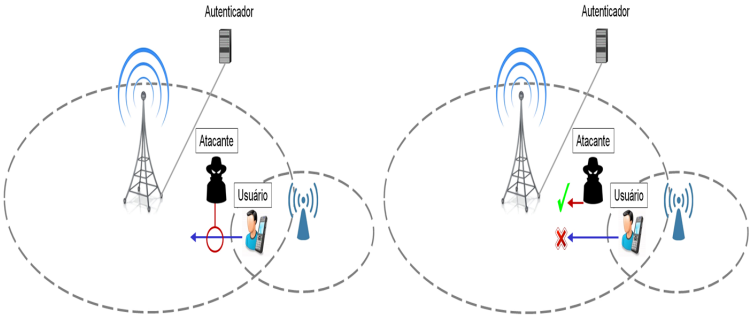


Figura 1. Ataque de repetição

A Figura 1 representa o ataque de repetição. No instante T1, o usuário do dispositivo móvel (UE) efetiva a autenticação durante o processo de *handover* vertical nas redes heterogêneas para obter acesso na rede representada pelo AP2. Um dispositivo malicioso (ATCK) sucede a captura dos quadros de transmissão no alcance do AP2. Logo, o UE, que esta em movimento, se autentica na rede destino e o ATCK efetua a captura. No instante T2, o UE se autentica com a rede de maior transmissão de dados sem fio através do AP1. O ATCK replica os quadros capturados de UE no AP2 e realiza a conexão com a rede para usufruir dos recursos e serviços oferecidos.

3.3. Esquema de autenticação CAH

O esquema de autenticação CAH baseia-se em uma autenticação unificada nas redes heterogêneas, na criptografia de pareamento e no mecanismo de proteção de ataques de repetição dos quadros de transmissão. A autenticação acontece em duas fases que são a inicialização e autenticação. Na fase de inicialização, o dispositivo móvel realiza a primeira autenticação para que a autenticação no serviço de *handover* seja realizada nas HetNets. Na fase de autenticação, o dispositivo móvel efetua a autenticação de *handover* e o mecanismo de proteção previne o ataque de repetição. Neste esquema, o ataque de repetição no serviço de autenticação de *handover* não surte efeito nas redes heterogêneas em virtude do mecanismo de proteção utilizar métodos de verificação tempo, verificação de quantidade de quadros recebidos na rede e informar as redes próximas sobre os dispositivos móveis.

3.3.1. Fase de inicialização

A fase de inicialização tem o objetivo de efetivar uma conexão segura para o usuário do dispositivo móvel. Esta autenticação acontece através de um gerenciador de autenticação central que auxilia o controle de acesso nas redes. O gerenciador central distribuí as identidades e as chaves para os dispositivos computacionais presentes nas redes heterogêneas. Desta forma, o dispositivo ao se autenticar com o gerenciador central ele obtém identidade e chaves que são utilizadas nas demais redes. O autenticador central se encontra no núcleo de processamento da rede de maior abrangência de sinal sem fio. A rede de maior abrangência integra as redes de menor poder de sinal sem fio. A integração de identidades e chaves tem relação com o a criptografia baseada em pareamento.

A Criptografia Baseada em pareamento (*Pairing Based Cryptography*) possibilita uma melhora de desempenho no serviço de transição. O funcionamento compreende de um grupo cíclico aditivo G e um grupo cíclico multiplicativo GT de mesma ordem q. O grupo G é um conjunto que possui as propriedades de identidade e associação. O parâmetro P é um gerador de arbitrário de G e aP que indica o P posicionado para ele mesmo. Um mapeamento bilinear e corresponde à $e = G * G \rightarrow GT$ que satisfaz as regras de bilinearidade e computabilidade. A bilinearidade representa a regra $e(aP, bQ) = e(P, Q)^a b$, onde $P, Q \in G$ e $a, b \in Z * q$. O valor $Z * q$ corresponde à $Z * q = \{\rho | 1 \leq \rho \leq q-1\}$. O mapeamento bilinear deve cumprir a regra $e(P, P) \neq 1$. A regra de computabilidade constitui-se de um algoritmo de eficiência que calcula $e(P, Q)$ para qualquer $P, Q \in G$.

A Figura 2 representa a fase de inicialização. O dispositivo móvel (UE) requisita uma conexão na rede de maior alcance. Primeiramente, o gerenciador central(KGC) efetua a autenticação do dispositivo. Através da autenticação,a identidade para o UE é gerada juntamente com as chaves pública e privada (chave assimétrica) para acesso entre as redes heterogêneas. A partir disso, o UE requisita autenticação na rede de menor alcance para criação de identidade e chaves. Neste caso, esta identidade e chaves devem ser armazenadas no gerenciador central das redes para ser associado a outros tipos de redes. A associação ocorre quando o UE requisitar em outro momento a autenticação com o outro tipo de rede através da relação entre as identidades e chaves.

Seja, G e GT de mesma ordem q e P um gerador randômico de G, $G * G \rightarrow GT$, cria-se um mapa bilinear. Assim, o gerenciador de chave central escolhe um número aleatório $s \in Z * q$ como a chave mestra e calcula a chave pública correspondente $pub = sP$. Além disso, duas funções hash de segurança $H1$ e $H2$ são determinadas, onde $H1 : \{0,1\}* \rightarrow G$ e $H2 : \{0,1\}* \rightarrow Z * q$. O gerenciador central publica os parâmetros de autenticação das redes $\{G, GT, q, P, pub, H1, H2\}$ e mantém a chave privada secretamente. Para cada ponto de acesso, o gerenciador de chaves central calcula $H1(IDAP)$ como a chave pública e $sH1(IDAP)$ como a chave privada. Desta forma, este gerenciador envia as chaves para o ponto de acesso (AP), onde $IDAP$ representa a identidade de cada AP.

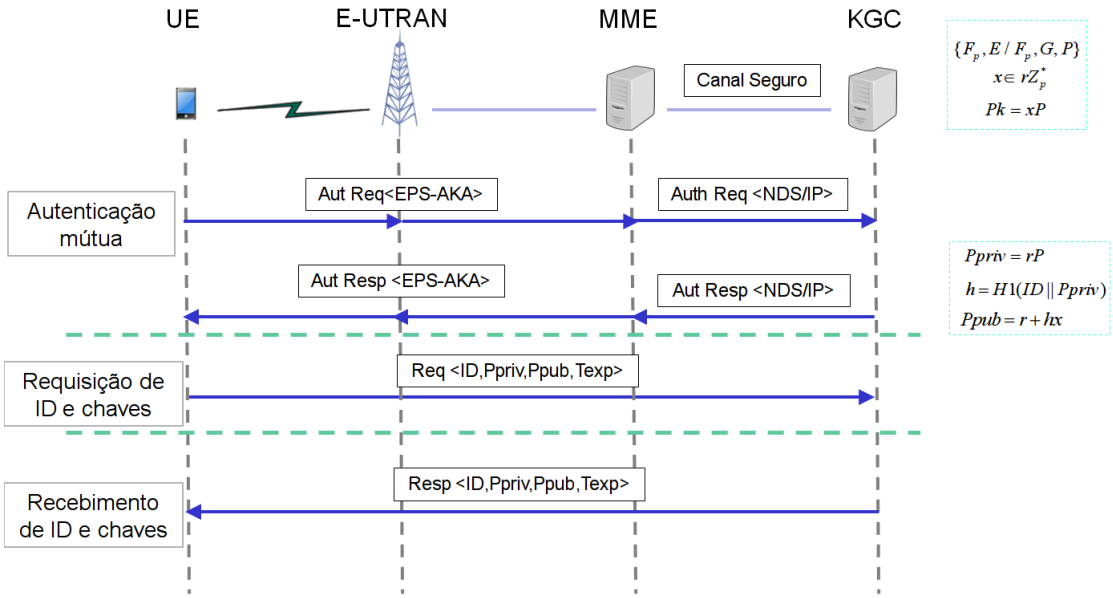


Figura 2. Inicialização

3.3.2. Fase de autenticação

A fase de autenticação no *handover* tem o propósito de prevenir o ataque de repetição durante o *handover* do dispositivo móvel. Nesta fase, o dispositivo móvel utiliza as chaves pública e privada para efetuar a autenticação durante o *handover* vertical entre redes heterogêneas. A chave pública e privada (assimétrica) auxilia a proteção contra o ataque de repetição durante a requisição do dispositivo móvel para o ponto de acesso.

A chave assimétrica garante que o dispositivo legítimo possa decifrar a mensagem de autenticação através da chave privada (chave secreta). Apesar disso, a captura de quadros acontece de acordo com as mensagens do dispositivo móvel e ponto de acesso. Para prevenir este ataque o mecanismo de proteção deve verificar os quadros. A Figura 3 mostra a fase de autenticação durante o handover. Um dispositivo móvel (*UE*) segue o protocolo de autenticação de entrega quando um ponto de acesso (*AP*) está em seu alcance de comunicação direta. O dispositivo móvel possui sua identidade e chaves criptografadas. A mensagem de autenticação $M = (pID|IDAP|ts)$ é enviada para o ponto de acesso, onde pID representa a pseudo-identidade do UE. O dispositivo móvel calcula a assinatura $AS = H2(M) * sH1(pID)$, onde um *timestamp* é adicionado para auxiliar a prevenção de ataques de repetição e a mensagem indica uma operação de concatenação. Desta forma, todas as entidades da rede preservam a sincronização de tempo de mensagem através de um mecanismo de sincronização de tempo. Em seguida, o UE envia mensagens de solicitação de acesso $\{Mi, A\}$ ao *AP*. Deste modo, o UE calcula a chave compartilhada simétrica com $AP : Ki - 2 = e(sH1(ipID), H1(IDAP))$.

Após a recepção de $\{M, A\}$, o AP efetiva os procedimentos de verificação através do mecanismo de proteção. O AP também efetua suas tarefas de resposta de autenticação. Os tempos são verificados para auxiliar na prevenção do ataque de repetição. O tempo de envio incluído no pID auxilia a verificação do tempo de expiração da mensagem do serviço de autenticação. Através da atribuição dos parâmetros re-

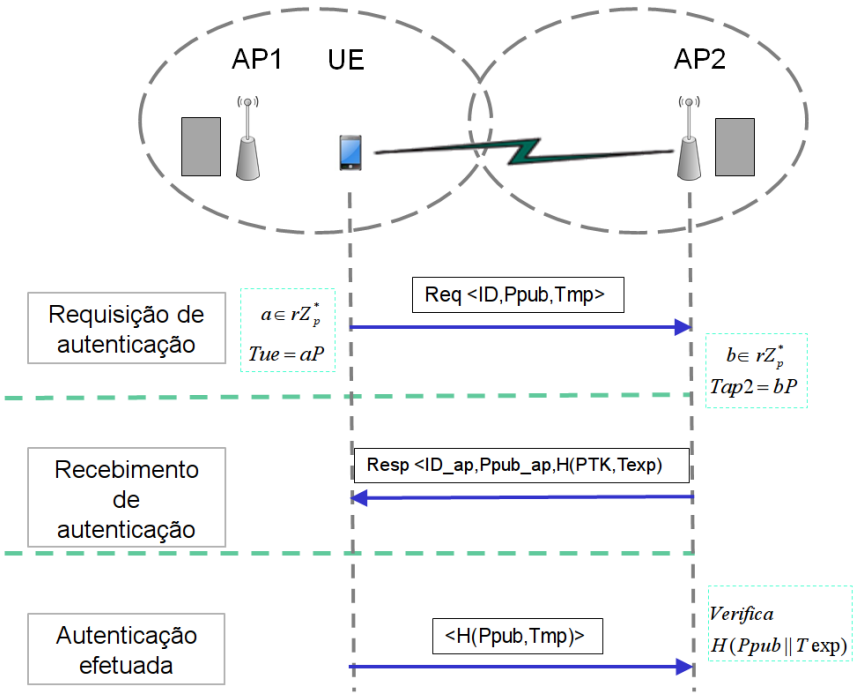


Figura 3. Autenticação

alizada pelo gerenciador central, o AP analisa se a assinatura A possui validade se $e(Ai, P) = e(H2(M) * H1(pID), pub)$. Então, $e(A, P) = e(H2(M) * sH1(pID), P)$ = $e(H2(M) * H1(pID), sP)$ = $e(H2(M) * H1(pID), pub)$.

O *AP* calcula ainda $K2 - i = e(H1(pID), sH1(IDAP))$. Os valores são analisados sendo $K2 - i = e(sH1(pID), H1(IDAP))$ = $e(H1(pID), H1(IDAP))$ $s = e(H1(pID), sH1(IDAP)) = K2 - i$. O AP gera um código de autenticação $Aut = H2(K - 2|pID|IDAP)$. Portanto, o AP envia as informações $\{pID, IDAP\}$ e *Aut* para o dispositivo móvel. Ao receber a mensagem $\{pID, IDAP, Aut\}$, dispositivo móvel gera um código de verificação $H2(Ki - 2|pID|IDAP)$ e compara com *Aut*. Se as informações $\{pID, IDAP\}$ e *Aut* corresponderem, o dispositivo móvel certifica-se que o AP é legítimo e estabeleceu uma chave *K* compartilhada 2-i; caso contrário, dispositivo móvel rejeita a conexão com o ponto de acesso.

3.3.3. Mecanismo de proteção

O mecanismo de proteção tem o objetivo de prevenir a personificação de identidade do ataque de repetição em redes que estejam no alcance do atacante no momento da captura de quadros. Este mecanismo atua no recebimento da mensagem de autenticação. A Figura 4 apresenta o mecanismo de proteção. Os módulos presentes no mecanismo de proteção são a verificação de tempo, verificação de quadros e atualização de conexão.

O módulo de verificação de tempo analisa o recebimento da mensagem *Trec*. Esta análise sucede da diferença do tempo de mensagem de autenticação enviada pelo usuário *Texp*, tal que *Texp* é embutida no quadro de autenticação, e o tempo de recebimento *Trec*. Desta forma, a diferença consiste de *Texp* - *Trec*. Este módulo fornece a garantia de que a

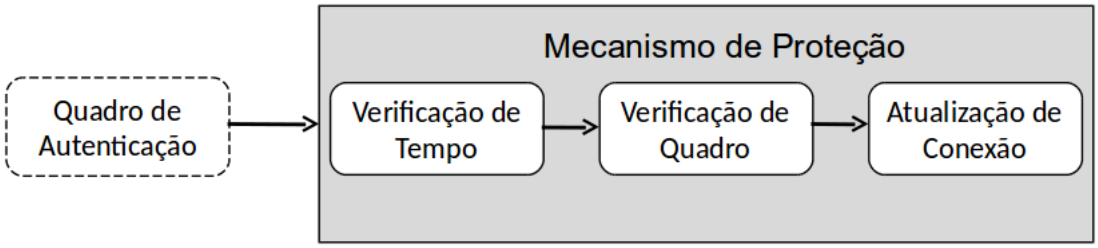


Figura 4. Mecanismo de Proteção

mensagem é única durante o periodo de tempo.

Cada quadro *qt* recebido pelo ponto de acesso detém sua identificação. A identificação é feita através da classificação de chegada de cada *qt*. Quando o quadro de autenticação *qt* chega no ponto de acesso, ele tem os campos analisados e classificado como *qt* = 1, e então outros quadros que possuam os mesmo campos de autenticação serão recusados. Esta estratégia invalida o ataque de repetição que visa a personificação de um dispositivo móvel. A estratégia também auxilia a caracterizar se este ataque esta sendo utilizado para efetuar negação de serviço. Para prevenir que o ataque seja efetuado em redes vizinhas, uma mensagem do ponto de acesso para os pontos de acesso vizinhos é enviada. Esta mensagem informa que o dispositivo móvel com uma identidade ID_i está conectado no ponto de acesso AP_i. Isso evita que o atacante propague a personificação em outras redes do ambiente sem fio.

O envio das mensagens ocorre entre as redes através da verificação do conjunto de redes vizinhas. As redes mais próximas serão informadas sobre a identidade e conexões de um dispositivo móvel. Um conjunto de vizinho *V* onde $V = \{ap_1, ap_2, ap_3, ..., ap_i\}$ tal que $ap_i \in V$. Para que a redes estejam completamente informadas sobre a conexão do usuário, o ponto de acesso envia a mensagem para o centro da rede de maior alcance, sendo a quantidade de mensagens *M* enviadas do AP para atualização de conexão representada por *M* onde $M = V + 1$. Este módulo informa os pontos de acesso vizinhos que o dispositivo móvel que possui uma identidade especifica esta conectado na rede.

4. Avaliação de Segurança e Desempenho

Esta seção apresenta a avaliação do esquema de controle de autenticação proposto para prevenir o ataque de repetição na autenticação do processo de *handover* em redes heterogêneas. Ele foi implementado no simulador NS-3, versão 3.24.1. O controle de autenticação foi avaliado levando em conta aspectos de segurança e desempenho da rede. Além disso, o esquema foi comparado com outra abordagem de autenticação chamada de UHA. A avaliação considerou um cenário composto por redes de diferentes tecnologias, dispositivos móveis e dispositivos maliciosos que executam o ataque de repetição.

O cenário utilizado na avaliação representa um ambiente de centro urbano com redes sobrepostas que possuem longo e curto alcance de sinal sem fio. Neste cenário representado pela Figura 5, os nós se movem no ambiente e utilizam o processo de transição de conexão nas redes sem fio. Logo, o serviço de autenticação prestará suporte ao processo de *handover* sempre quando for necessário a transição para outra rede. Os nós atacantes posicionam-se em partes do cenário onde existe grande sobreposição de redes. Desta forma, estes nós efetivam uma grande captura de quadros dispositivos móveis do

usuários legítimos presentes no cenário durante a autenticação na transição.

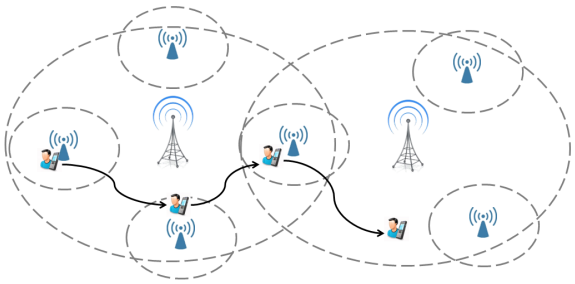


Figura 5. Cenário de Avaliação

Parâmetros	Valores
Quantidade de nós	20,40 e 60
Modelo de mobilidade	Random Waypoint
Velocidade dos nós	0,2m/s a 2m/s
Tempo de Simulação	600 s
Tecnologia	WiFi e LTE
Nós atacantes	10%

Tabela 1. Parâmet. de Simulação

A Tabela 1 dos parâmetros de simulação resume os valores usados na configuração das redes heterogêneas. O parâmetro da quantidade de nós na avaliação variou entre 20, 40 e 60. Os nós que compreendem os dispositivos móveis comum possuem mobilidade. Esta mobilidade é aleatória no cenário de avaliação com velocidades entre 2m/s e 5m/s. Os nós atacantes são fixos. A posição destes nós é aleatória em relação as localizações do cenário onde existem sobreposições de redes. A comunicação dos nós legítimos considera as tecnologias com padrões 802.11 (WIFI) e Long Term Evolution (LTE). A simulação foi repetida 30 vezes e cada simulação teve a duração de 600 segundos. A quantidade de atacantes representa 10% da quantidade de nós da avaliação.

As métricas para avaliar o controle de autenticação compreendem os aspectos de segurança e desempenho da rede. Os aspectos de segurança que relacionam o tipo do ataque consistem da taxa de detecção, taxa de falso positivo e taxa de falso negativo. A taxa de falso negativo não é levada para o contexto de avaliação pois não existem ocorrências para averiguar a métrica. As métricas de segurança usadas são a ***Taxa de ataques de repetição prevenidos*** (T_{prev}) e a ***Taxa de falso positivos*** (T_{fa}). A métrica T_{prev} representa os quadros repetidos descartados pelos esquemas que corresponde a razão entre o somatório de ataques prevenidos, *prev*, e a quantidade de ataques de repetição, *rep*, (Eq. 1). A métrica T_{fa} corresponde a quantidade de vezes que os esquemas identificaram um ataque quando não existia e é representado pelo somatório de prevenções, *prev*, pelo total de requisições de autenticação validas *req*, (Eq. 2).

$$T_{prev} = \frac{\sum prev}{rep} \quad (1)$$

$$T_{fa} = \frac{\sum prev}{req} \quad (2)$$

$$T_x = \frac{\sum T_{UE-AP}}{Q_{aut}} \quad (3)$$

$$T_y = \frac{\sum T_{AP-AP}}{msg} \quad (4)$$

$$T_z = \frac{\sum T_{AP-A}}{req} \quad (5)$$

$$C_{ini} = \frac{\sum T_{ini}}{Q_{ini}} \quad (6)$$

$$C_{aut} = \frac{\sum T_{aut}}{Q_{aut}} \quad (7)$$

As métricas de desempenho são ***Tempo médio de transmissão entre dispositivo móvel e ponto de acesso*** (T_x), ***Tempo médio de transmissão entre pontos de acesso*** (T_y), ***Tempo médio de transmissão entre ponto de acesso e autenticador*** (T_z), ***Custo computacional da inicialização*** (C_{ini}) e ***Custo computacional da autenticação*** (C_{aut}).

A métrica T_x consiste do somatório de tempo da transmissão entre o dispositivo móvel UE e o ponto de acesso AP, T_{UE-AP} e a quantidade de autenticações efetuadas Q_{aut}, (Eq. 3). A métrica T_y compreende o somatório de tempo da transmissão os pontos de acesso, T_{AP-AP} e a quantidade de mensagens enviadas msg, (Eq. 4). A métrica T_z compreende o somatório de tempo da transmissão entre o ponto de acesso, AP, e o autenticador A, T_{AP-A} em relação à quantidade de autenticações requisitadas entre eles req,(Eq. 5).

Já a métrica C_{ini} abrange a razão do tempo de todas as inicializações, T_{ini}, pela quantidade de inicializações efetuadas, Q_{ini}, (Eq. 6). A métrica C_{aut} representa o somatório do tempo de autenticação, T_{aut}, pela quantidade de autenticações, C_{aut}, (Eq. 7). Estas métricas de desempenho consistem do tempo em milissegundos gasto para concluir as tarefas tanto da transmissão como também o tempo para executar cada fase do esquema de autenticação.

4.1. Resultados

O Gráfico 6 mostra a taxa dos ataques de repetição prevenidos durante a simulação de 600 segundos. O esquema concorrente demonstra ser mais vulnerável por não possuir uma prevenção completa contra o ataque de repetição. Isto acontece em virtude da prevenção ser executada somente no mesmo ponto de acesso em que o nó legítimo está transitando. O controle de autenticação proposto (CAH) previne o ataque de repetição de forma mais abrangente, ou seja, levando em conta todas as possibilidades de ataque.

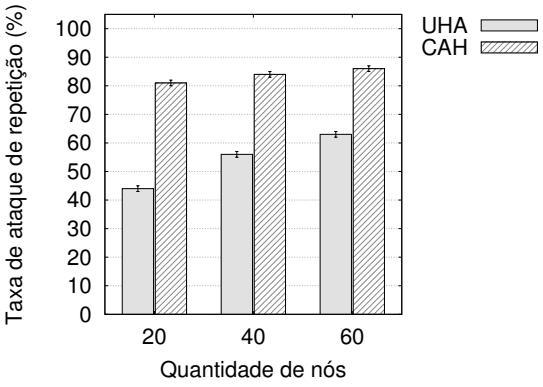


Figura 6. Taxa de prevenção

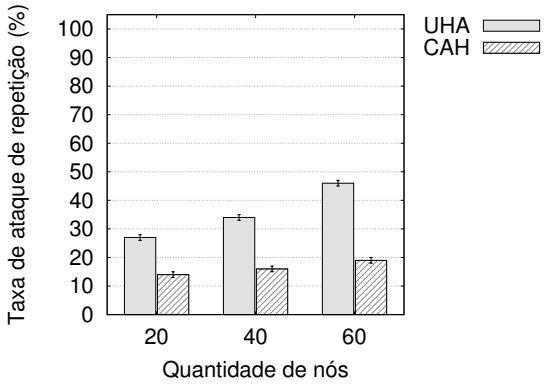


Figura 7. Taxa de falsos positivos

O Gráfico 7 mostra a comparação da taxa de falsos positivos. O esquema UHA apesar de não levar em conta todos os casos possíveis deste ataque, possui uma taxa de

falso positivo maior que o CAH. Isso ocorre em virtude dos outros ataques serem prevenidos de forma mais eficaz pelo mecanismo de proteção do CAH que leva em consideração o tempo e quantidade de mensagem. O componente de informação deste mecanismo consegue eliminar os ataques nas redes vizinhas de forma eficaz.

O Gráfico 8 apresenta a taxa total de ataques contra os tipos de tecnologias de redes. Esta diferença ocorre devido à característica de comunicação de cada tipo de rede ser diferente e pela variação de mobilidade dos dispositivos móveis. Isto proporciona aos dispositivos maliciosos uma maior capacidade para efetuar ataques. O gráfico 9 expõe a taxa total de ataques prevenidos em cada tipo de tecnologia.

A diferença de resultados compreende as características propostas por cada esquema de autenticação. O esquema UHA possui um tratamento de mensagens de autenticação efetivo em redes WiFi enquanto o esquem CAH impede ataques nas duas tecnologias. Isto ocorre pelo motivo da ausência de um tratamento do UHA em mensagens de ambas as tecnologias permitindo que o atacante obtenha o acesso pela identidade na rede de maior alcance e com isso usufruir dos serviços.

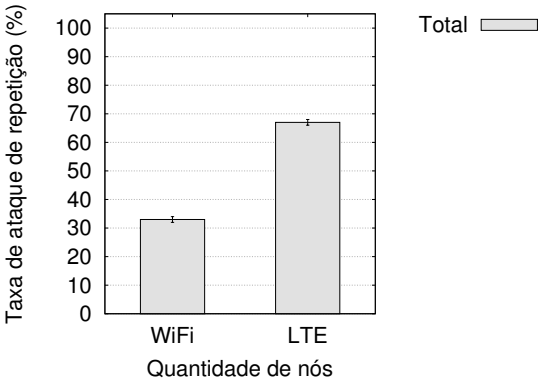


Figura 8. Taxa de ataques por tecnologia de rede

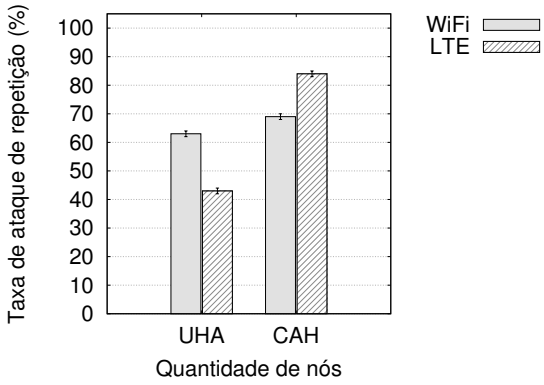


Figura 9. Taxa de prevenção por tecnologia de rede

Para avaliar o desempenho do esquema CAH com o UHA foi levado em conta a comparação da transmissão das mensagens e o custo computacional. O custo de transmissão da mensagem de autenticação consiste da comunicação entre o UE e o AP, o custo entre APs e entre o AP e o servidor de autenticação (AU), respectivamente. Desta forma, a Tabela 2 mostra as despesas gerais de transmissão dos sistemas de comunicação, e representa o desempenho das comunicações entre as entidades da rede heterogênea. As comunicações consistem do dispositivo móvel e ponto de acesso (UE-AP), do ponto de acesso com ponto de acesso (AP-AP), e do ponto de acesso e autenticador central (AP-A). A diferença de tempo está na relação da comunicação entre o dispositivo móvel e o ponto de acesso devido ao uso do pareamento que representa um método de processamento rápido para cifrar a informação.

Esquema	UE-AP	AP-AP	AP - A
UHA	4 ms	0	2 ms
CAH	2 ms	0	2 ms

Tabela 2. Comparação de transmissão

Esquema	Tempo de Inicialização	Tempo de *handover*
UHA	3 ms	4.5 ms
CAH	3 ms	2 ms

Tabela 3. Comparação de custo computacional

A Tabela 3 mostra uma comparação do custo computacional do esquema de autenticação UHA com o CAH. O pareamento obteve uma autenticação mais rápida em relação a troca de mensagens. O controle de autenticação CAH mostrou-se mais seguro e com melhor desempenho de comunicação em relação ao concorrente UHA. Isto se deve pelo motivo do pareamento ser realizado de forma unificada entre as redes possibilitando ao serviço de autenticação uma execução mais rápida.

5. Conclusão

Este trabalho apresentou o controle de autenticação (CAH) que possibilita uma melhor gerência dos dispositivos móveis nas redes heterogêneas e uma prevenção do ataque de repetição mais robusta no serviço de handover. Este controle consiste de um esquema de autenticação composto por duas fases e um mecanismo de proteção. As fases compreendem a autenticação inicial dos dispositivos móveis com a rede de maior alcance de transmissão sem fio e a autenticação no serviço de *handover* nas redes heterogêneas. O mecanismo realiza a proteção das informações de autenticação durante o *handover* e informa as redes vizinhas sobre os dispositivos autenticados. Para isso, o mecanismo executa a análise do tempo e da quantidade dos quadros de transmissão da autenticação recebidos na rede, e informa às redes vizinhas sobre a autenticação do dispositivo móvel com o propósito de evitar ataques e melhorar o desempenho da autenticação no serviço de *handover*. O controle de autenticação foi avaliado através do simulador NS-3 e a biblioteca crypto++. As simulações demonstraram que o controle de autenticação proposto através do uso de um esquema unificado de autenticação permite a prevenção de ataques de captura e melhora o desempenho do *handover*. Trabalhos futuros envolvem a análise da autenticação em relação à outros tipos de ataques que afetam a segurança das redes heterogêneas.

Referências

Cao, J., Ma, M., and Li, H. (2012). Unified handover authentication between heterogeneous access systems in lte networks. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 5308–5313. IEEE.

Cao, J., Ma, M., Li, H., Zhang, Y., and Luo, Z. (2014). A survey on security aspects for lte and lte-a networks. *Communications Surveys & Tutorials, IEEE*, 16(1):283–302.

Cheng, S.-M., Ho, C.-H., Chen, S., and Chang, S.-H. (2014). Distributed anonymous authentication in heterogeneous networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*, pages 505–510.

Cremers, C. (2008). On the protocol composition logic pcl. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 66–76. ACM.

Damnjanovic, A., Montojo, J., Wei, Y., Ji, T., Luo, T., Vajapeyam, M., Yoo, T., Song, O., and Malladi, D. (2011). A survey on 3gpp heterogeneous networks. *Wireless Communications, IEEE*, 18(3):10–21.

Feng, T. and Jiao, J. (2012). WAPI secure access authentication scheme for heterogeneous networks based on identity-based cryptograph. In *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, volume 1, pages 130–135. IEEE.

Fu, A., Zhang, G., Zhang, Y., and Zhu, Z. (2013). Ghap: An efficient group-based handover authentication mechanism for ieee 802.16 m networks. *Wireless personal communications*, 70(4):1793–1810.

He, D., Chan, S., and Guizani, M. (2015). Handover authentication for mobile networks: security and efficiency aspects. *Network, IEEE*, 29(3):96–103.

He, D., Chen, C., Chan, S., and Bu, J. (2012). Secure and efficient handover authentication based on bilinear pairing functions. *Wireless Communications, IEEE Transactions on*, 11(1):48–53.

Heath, R., Kountouris, M., and Bai, T. (2013). Modeling heterogeneous network interference using poisson point processes. *Signal Processing, IEEE Transactions on*, 61(16):4114–4126.

Hu, Y.-C., Perrig, A., and Johnson, D. (2003). Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1976–1986 vol.3.

Kent, A. D., Liebrock, L. M., and Neil, J. C. (2015). Authentication graphs: Analyzing user behavior within an enterprise network. *Computers & Security*, 48(0):150 – 166.

Li, X., Zhang, Y., Liu, X., Cao, J., and Zhao, Q. (2012). A lightweight roaming authentication protocol for anonymous wireless communication. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 1029–1034. IEEE.

Ranjan, S., Akhtar, N., Mehta, M., and Karandikar, A. (2014). User-based integrated offloading approach for 3GPP LTE-WLAN network. In *Communications (NCC), 2014 Twentieth National Conference on*, pages 1–6.

Sandhu, R. and Samarati, P. (1994). Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48.

Xenakis, D., Passas, N., Merakos, L., and Verikoukis, C. (2014). Mobility management for femtocells in lte-advanced: Key aspects and survey of handover decision algorithms. *Communications Surveys Tutorials, IEEE*, 16(1):64–91.

Yu, Q., Jiang, W., and Xiao, Z. (2013). 3g and wlan heterogeneous network handover based on the location information. In *Communications, Circuits and Systems (ICCCAS), 2013 International Conference on*, volume 2, pages 50–54. IEEE.

Energy-efficient Adaptive Encryption for Wireless Visual Sensor Networks

Danilo de Oliveira Gonçalves[1], **Daniel G. Costa**[1]

[1]PGCA-UEFS, State University of Feira de Santana, Brazil

daniloxm@gmail.com, danielgcosta@uefs.br

Abstract. *Wireless sensor networks are usually composed of small sensor nodes with low processing power, limited memory and restricted energy supply. Among them, camera-enabled sensors can be used to gather visual data, but some relevant processing and transmission challenges are raised. In general, sensor networks have many vulnerabilities that can be exploited by attackers, demanding defense measures. However, traditional security mechanisms may impose high computing and communication overhead, which may compromise network performance specially when visual data is sensed from monitored fields. In this context, this paper proposes a new paradigm to ensure energy-efficient security for wireless visual sensor networks, defining an adaptive encryption approach. Doing so, confidentiality, authenticity and integrity are assured adaptively and according to application requirements, saving resources of networks while providing acceptable levels of protection for retrieved data.*

1. Introduction

Wireless Sensor Networks (WSN) are an emerging low-cost technology designed to retrieve information from several types of environments. Those networks are composed of hundreds, even thousands, of small devices with sensing, processing and communication capabilities [Yick et al. 2008]. Usually, WSN are only able to collect scalar data such as temperature, pressure or humidity. However, there is another type of sensor network that employs camera-enabled sensors to also retrieve video streams and still images [Costa and Guedes 2011], potentially enriching monitoring and control applications. Such Wireless Visual Sensor Networks (WVSN) retrieve much more information from the environment when compared to scalar sensor networks, imposing many designing and operation challenges.

WSN are inherently vulnerable to various types of security threats due to their distributed nature and the way they are employed, especially in remote areas [Yick et al. 2008]. In addition, wireless communications are subject to failures and security attacks. Depending on applications requirements, security aspects may be very important to ensure proper network operation, correct energy expenditure and the preservation of secrets. Therefore, security may be a very important issue in WSN design, but due to resource constraints in sensor nodes, traditional security mechanisms with high computing and communication overhead become too degrading for WSN [Sen 2009]. In short, security in WSN is a challenging task and resource limitations are even more stringent when networks retrieve multimedia data.

Generally speaking, wireless sensor networks are intrinsically vulnerable to internal and external attacks [Yick et al. 2008]. We then propose an innovative approach to

provide security to wireless visual sensor networks with small additional overhead. A new security paradigm is proposed herein, referred as Adaptive Encryption, which is centered at minimizing the use of network resources to provide security.

The proposed adaptive encryption paradigm arises as a generic security approach that is premised on the adaptation of security mechanisms according to application monitoring requirements. This paradigm is centered at the definition of Confidential Areas (CA), which define an area of influence and an expected level of confidentiality. Actually, a wireless visual sensor network may be monitoring areas of interest with different demands for confidentiality. The network then employs a specific encryption mechanism that optimally provides protection of sensed data. For example, a wireless visual sensor network monitoring a factory may view areas with different relevances for an application, as the parking area, the engines rooms, entrance halls and the manager office. Each of these areas may have different confidentiality requirements concerning critical data that have to be protected from external attackers. In other words, using this model, it is not necessary to apply security mechanisms for the network as a whole, if the application requirements indicate that only a few sensor nodes in specific areas should have their data secured.

The proposed approach aims to minimize the use of network resources that are required to provide security, mainly energy consumption, when compared to mechanisms that provide the same protection for the entire network. As most wireless sensor networks have limitations in communication, processing, memory, storage and energy, the proposed model can be valid to aggregate different security levels without degrading network performance. Besides Confidential Areas, we also define procedures to create those areas and a specialized protocol. Moreover, we define different security schemes for validation of the proposed approach.

The remainder of the paper is organized as follows. The second section presents some related works. Section 3 presents the key definitions of the proposed approach. Section 4 defines the procedures to configure confidential areas. Initial results are presented in Section 5, followed by conclusions and references.

2. Related works

Many applications for wireless visual sensor networks may require minimum levels of security to assure confidentiality, authenticity and integrity for images captured by a network. However, traditional security mechanisms are not adapted to such networks. Thus, there is a need for optimizations and new paradigms to efficiently provide security in WVSN.

Attacks will typically be centered in exploiting vulnerabilities in some communication layer, eavesdropping transmitted data, altering confidential data or prejudicing the network operation with artificial malicious information [Gonçalves and Costa 2015]. Security threats may be of different types and may have different impacts on wireless sensor networks [Wang et al. 2013, Winkler and Rinner 2014, Chen et al. 2009]. This complex scenario push us to incorporate some defense mechanisms, which should comply with particularities and limitations of employed sensor nodes.

Cryptography is the basic defense mechanism in wireless sensor networks. It may be employed to provide authenticity, confidentiality and integrity for sensed data.

Actually, all communications can be intercepted by malicious nodes, demanding some protection. However, cryptography may be too resource-demanding, potentially degrading the limited resources of sensor networks. In this context, selective encryption comes as an optimized method that provides a reasonable level of secure data transmission with reduced overhead [Lecuire et al. 2007, Costa and Guedes 2011]. In short, this principle exploits characteristics of media coding algorithms to provide secrecy while reducing computational complexity [Grangetto et al. 2006].

In selective encryption, the basic idea is to encode only a set of blocks of sensed images. The work in [Nikolakopoulos and Fanakis 2009] proposes an adaptive compression mechanism to adapt transmission rate according to current network conditions. In the same way, the work in [Nikolakopoulos et al. 2010] proposes the use of Quadtree decomposition to support adaptive image compression and efficient congestion control. Raw images are decomposed using a Quadtree algorithm, and authors propose that higher compression should be adopted when fewer information should be transmitted, in a dynamic and adaptive way. The work in [Wang et al. 2010] proposes a selective encryption mechanism for video streams encoded using MPEG-4 codec. The relevance of video frames for the reconstruction process at the destination is exploited, where only high-relevant parts are encrypted.

In a similar context, data aggregation combines and summarizes data packets from several sensor nodes, which may be then encrypted [Ozdemir and Xiao 2009]. As source nodes may be compromised, authentication mechanisms may be also required before aggregation. A malicious node may provide false data packets that may reduce quality of aggregated data [Simplicio et al. 2013]. The work in [Gao et al. 2014] exploits the similarity that may be present in multimedia data, compressing sensed data for transmission. In a different way, the work in [Elsabi and Ozdemir 2012] combines data aggregation with watermarking, considering aggregation of scalar and image data.

All these security approaches influenced our work, but we proposed a different innovative solution for security protection. The definition of confidential areas, which can be dynamically created over monitored fields, is proposed in this paper to guide optimized cryptography, potentially bringing significant results for WVSN. And so cryptography context is not defined for specific transmission flows, as it happens when selective encryption is employed. Actually, the proposed approach defines a global perception of confidentiality that is valid for the entire network. To the best of our knowledge, no work has proposed such approach before.

3. Proposed approach

The idea behind the proposed approach of adaptive encryption is to provide security for wireless visual sensor networks without severely degrading the limited resources of sensor nodes. In order to achieve these objectives, the adaptive encryption model is centered at the concept of *Confidential Area* (CA). In addition, we propose herein other concepts related to confidential areas, as *Confidentiality Levels* and *Security Schemes*. These three concepts together will govern the operation of the proposed adaptive encryption model.

Sensing applications may require different confidentiality levels for certain parts of any monitored area. Thus, using the proposed paradigm, a sensor network is divided into smaller areas, where these areas may have different security levels that can be re-

flected in specific security mechanisms. Doing so, we can provide strong security only to areas with high security requirements, leaving the remaining areas with weaker security protection.

3.1. Confidential areas

The Confidential Area (CA) is a concept that defines a delimited geometrical area that is associated to a particular confidentiality level. Each confidentiality level is then mapped to a security scheme, which defines how data collected by sensor nodes will be protected. In other words, sensors included into a confidential area will be subject to a security pattern defined by applications. A typical WSN can have more than one CA, each one with particular levels of confidentiality. The application requirements concerning security demands for retrieved data and the characteristics of the monitored field will guide the definitions of confidential areas in sensor networks.

Therefore, the concept of CA defines that each delimited area has a particular confidentiality level, but how to provide security for collected data inside each area varies according to the adopted security scheme. Actually, although it is defined as a confidential area, the use of cryptography algorithms will naturally provide also authenticity and integrity for transmitted data. The use of the term "confidentiality" was to emphasize the critical nature of some of retrieved data.

We defined a set of characteristics for Confidential Areas in wireless visual sensor networks, as follows:

- The creation of CAs and the inclusion of sensor nodes into them is performed centrally, at the sink side;
- Confidential areas do not overlap. That is, there is no intersection among different confidential areas, neither between areas with the same confidentiality levels;
- It may change over time. The borders of any CA may be adjusted, or the confidentiality level may be updated;
- A CA will always be a convex 2D quadrilateral. In this way, a CA is defined by a fivefold: CA(N,P,Q,R,S), where N is its confidentiality level and P, Q, R e S are ordered pairs of vertices of the quadrilateral.
- Some location service must be employed to support the proposed approach, which may be based on reference points, relative positions or GPS (Global Positioning System) coordinates [Pescaru and Curiac 2014];
- The definitions of all CAs in a WVSN are strictly indicated by application requirements, which is an abstract concept. The same physical network may have different CAs, according to the defined monitoring tasks;
- The concept of confidential areas is not used to prioritize data traffic generated within a CA. However, confidential data may also require QoS guarantees;

3.2. Confidentiality levels

The concept of Confidentiality Level (CL) is used to numerically represent a need for confidentiality for data originated from the corresponding CA. We define four different values for CL, as follows:

- **Level 0:** Area without encryption or any security mechanism, being the lowest level. This level is not associated to any CA, since it exists only to configure nodes that are not inside any CA. The notation is $\text{CA}_{N=0}$;

- **Level 1:** CA with low security. In those areas, sensor nodes require light security, with "low" protection being applied. The notation is $CA_{N=1}$;
- **Level 2:** CA with moderate security requirements. Security at this level is slightly higher than the previous level, but with features that ensure some resource savings. Its notation is $CA_{N=2}$;
- **Level 3:** CA with maximum security. At this level, all collected data is protected at the highest level, even with high use of sensors' resources. The notation is $CA_{N=3}$.

Often, there may be the need for more than one CA with the same value for CL. An example is the use of this model in an intrusion detection system, where the doors of a building can be monitored by sensor nodes inside multiple $CA_{N=3}$. In this case, there will be several CAs with level 3 on entering and exiting doors of the building.

3.3. Security schemes

The security schemes define which mechanism, aspect or security measure will be applied to each CA depending on their confidentiality levels. Such schemes may define, for example, encryption algorithms, key sizes, key management approaches or other mechanisms associated to data protection.

A wireless visual sensor network may define any type of security scheme, according to its monitoring requirements, but usually an unique scheme will be employed for the entire network at a time. We created some reference schemes to be used in generic networks, presented as follows, but generally any security scheme may be defined.

- **Scheme 1 - Coding**:
  - Level 0: Unencrypted;
  - Level 1: Selective image encryption (DWT at two levels);
  - Level 2: Selective image encryption (DWT at one level);
  - Level 3: Full image encryption.
- **Scheme 2 - Encryption key size**:
  - Level 0: Unencrypted;
  - Level 1: 128-bit key;
  - Level 2: 192-bit key;
  - Level 3: 256-bit key.
- **Scheme 3 - Collected data type**:
  - Level 0: Unencrypted;
  - Level 1: Encryption of only scalar data;
  - Level 2: Encryption of scalar data and still images;
  - Level 3: Encryption of scalar data, still images and video streams.

Therefore, a security scheme will define which action will be taken when applying the proposed adaptive encryption model. In addition, it is worth mentioning that data collected inside a CA level 1 still has some confidentiality, even with reduced protection. Actually, the concept of security scheme makes it clearer that the general idea of the proposed approach is to degrade less network resources when compared to security approaches that protect the network as a whole.

4. Network deployment and configuration

One key issue of the proposed approach is how to associate deployed sensor nodes to defined confidential areas. As CAs are defined by applications, nodes must know which CA they belong to.

In wireless visual sensor networks, camera-enabled sensors monitor an area based on a direction of viewing, defined by their Field of View (FoV) [Costa and Guedes 2011]. Hence, the same sensor may view areas "belonging" to different CAs. Actually, there are different configurations for sensor nodes in relation to a confidential area, concerning the overlapping of the sensor's FoV and the defined CA.

We define that only for the configuration where a sensor node and its field of view are both outside of a considered CA, that sensor node is assumed as not belonging to that confidential area. In all other cases the sensor node belongs to the considered CA.

Figure 1 presents the Field of View of camera-enabled sensors. For simplification, we assume the FoV as an isosceles triangle, defined by a sensing radius (R), a viewing angle (θ) and an orientation (α).

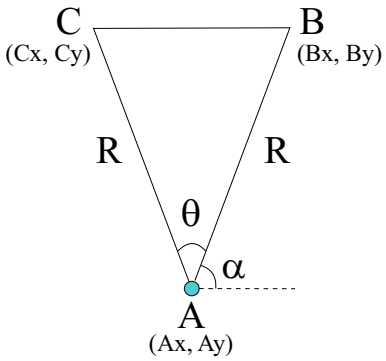


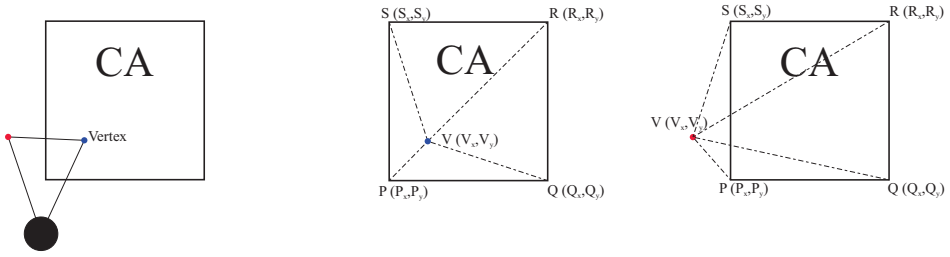
Figure 1. Field of View (FoV) of a visual sensor node [Costa et al. 2014].

4.1. Geometric model

We propose a geometric model to regulate the association of any sensor node to confidential areas. In this geometric model, each vertex is analyzed at a time. If at least one vertex of the FoV is within a CA (Figure 2(a)), the sensor node as a whole will be considered included in that CA, assuming its preset policies and security schemes. In order to perform the desired computing for each vertex, we employ linear geometry, as graphically expressed in Figures 2(b) and 2(c).

Assuming a quadrilateral PQRS which defines a CA and one vertex V, as shown in Figure 2(b), the following triangles will be created: VPQ, VQR, VRS, VPS. The formulation in (1) is employed to compute the areas of these triangles, comparing the result to the computed area of the CA.

$$\begin{aligned} &X = \Delta_{VPQ} + \Delta_{VQR} + \Delta_{VRS} + \Delta_{VPS} \\ &If\ X = \Delta_{PQRS}\ \ (Vertex\ within\ the\ CA) \\ &If\ X > \Delta_{PQRS}\ \ (Vertex\ out\ of\ the\ CA) \end{aligned} \tag{1}$$



(a) (b) (c)

Figure 2. Viewing a CA (a) Sensor node with one of the vertices within the CA; (b) Example of an included vertex; (c) Example of a not included vertex.

The areas of the created triangles will be computed as expressed in (2).

$$\begin{aligned}
\Delta_{VPQ} &= \frac{V_x.(P_y - Q_y) + P_x.(Q_y - V_y) + Q_x.(V_y - P_y)}{2} \\
\Delta_{VQR} &= \frac{V_x.(Q_y - R_y) + Q_x.(R_y - V_y) + R_x.(V_y - Q_y)}{2} \\
\Delta_{VRS} &= \frac{V_x.(Q_y - S_y) + R_x.(S_y - V_y) + S_x.(V_y - R_y)}{2} \\
\Delta_{VPS} &= \frac{V_x.(P_y - S_y) + P_x.(S_y - V_y) + S_x.(V_y - P_y)}{2}
\end{aligned} \tag{2}$$

Besides the computing of vertices positions, there are other scenarios that must to be treated. If a vertex is on top of one side of the CA or upon the prolonged line of one side of the CA, only three triangles can be created. This way, even when one of the areas of the four triangles is zero, if the sum of the other three areas is equal to the area of the CA, it means that the vertex is located over one side of the CA. If the sum of the three other areas of the triangles is greater than the area of the CA, it means that the vertex is on the prolonged line of one side of the CA and thus out of the CA.

Another situation is if one of the vertices of the triangle formed by the sensor node and its FoV is coincident with one of the vertices of the quadrilateral that delimits the CA. In this case, a coincidence test must to be performed. If the test is successful, the vertex is assumed as belonging to the considered CA.

A final remark is when the sensor's FoV covers two or more different CAs. In this case, the proposed approach associates the sensor to the CA with highest confidentiality level, in order to not compromise the application requirements.

4.2. Association protocol

The procedures previously described are required to associate sensor nodes to confidential areas. Actually, such computing is performed in a centralized way, at the sink side. Thus, there should be adopted some mechanism to inform sensor nodes that they belong to a particular confidential area, and hence they must follow a particular security scheme. Moreover, as it is an adaptive approach, changes in parameters of CA must be reflected into the network. For these cases, we propose the application-level Confidential Area Association Protocol (CAAP).

Initially, sensor nodes are pre-configured with the security scheme to be adopted by the sensing application. Each sensor node should already know what to do with sensed data when it is associated to a CA. Moreover, when the network starts operation, every sensor node is automatically associated to the confidential area with level 0.

The CAAP protocol has its operation centered at three different types of messages. A request/identification message of sensor node (CA-Request), a configuration message (CA-Configure) and an acknowledgment message (CA-ACK). CA-Request is a message that is only sent by the sink toward a sensor node. This message requests information about the sensor node, such as CA current level and location. In an ideal scenario where sensors do not change position, this message may become unnecessary if the sink node is configured with information of all sensor nodes. However, as position of sensor nodes may change along the time, the sink should send a CA-Request message requesting information of the sensor nodes. After receiving a CA-Request, the sensor node answers this message sending back a CA-ACK message for acknowledgment purposes. This CA-ACK message also reports the requested information, which is usually the current location and confidentiality level. The CA-Configure message is a return message, sent only by the sink node toward a sensor node and providing configuration after computing the corresponding CA. Finally, another CA-ACK message is also used for the acknowledgment of the CA-Configure message, finishing the communication cycle.

CAAP is intended to be an energy-efficient reliable application-layer protocol. As wireless transmissions are very susceptible to transmission errors and packet losses, this protocol provides retransmissions as an error recovery mechanism. As described previously, the CA-ACK message is sent only by a sensor node as a response and acknowledgment to other messages. Actually, the CA-ACK message is not confirmed by its receptor, i.e., it is used for the acknowledgment of two other messages, but who sends a CA-ACK does not receive any acknowledgment of receipt. This is due to energy constraints of sensor nodes.

The protocol operation is controlled by two counters: t_m and t_{ack}. The counter t_m sets the time that the sink node should wait for the receiving of a CA-ACK message. On the other hand, the counter t_{ack} sets the time required to ensure that a CA-ACK message was received by the sink node. Thereby, t_{ack} is used by sensor nodes to control whether it should assume or not that a transmitted CA-ACK message sent by it was correctly received by the destination. In such way, if no CA-ACK message is received by the sink before t_m, the corresponding message should be retransmitted. Actually, a retransmission will be required in the following cases:

- If a CA-request or CA-Configure message is lost (dropped or corrupted) during transmission;
- If a CA-ACK message is lost;
- If a CA-ACK message is received after t_m.

In order to ensure some scalability to the proposed protocol, t_m should double if a CA-request or CA-Configure message needs to be retransmitted. The maximum number of retransmissions of the same message was settled in 4, in order to avoid many retransmission attempts when routes are congested or inactive. Thus, for r as an attempt of retransmission, r = 1, ..., 4, and the end-to-end time for transmission, processing and ac-

knowledgment defined as t_r, we can define t_m as $t_m = 2^{(r-1)}.t_r$. The value for t_m is reset to the reference value t_r after each successful transmission.

It is expected that $t_{ack} < t_m$, even in cases of retransmission when t_m increases. Concerning counter t_{ack}, after sending a CA-ACK message, the transmitting sensor node waits for t_{ack} with the following constraints:

- If a CA-request or CA-Configure message is received before t_{ack}, it is assumed that the CA-ACK was lost;
- If a CA-request or CA-Configure message is receive after t_{ack}, it must be assumed that the CA-ACK was received correctly and that it is a new message.

Figure 3 illustrates an example of CAAP operation.

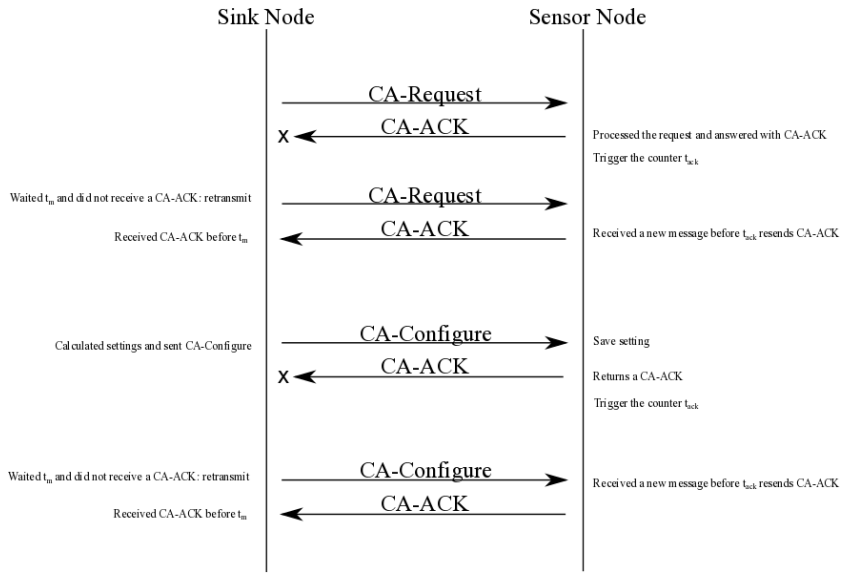


Figure 3. Operation example of CAAP protocol with CA-ACK losses.

5. Numerical results

In order to validate the proposed approach and attest the expected benefits, a series of mathematical verifications were performed. Some verification scenarios were defined, where AES (Advanced Encryption Standard) [Wang et al. 2011] was employed as the reference algorithm (although other algorithms could be employed with no prejudice to the performed verifications). We defined two security schemes: Scheme 1, based on DWT (Discrete Wavelet Transform) image coding, and Scheme 2, based on different AES key sizes, as previously defined.

Figure 4 presents a diagram for the first security scheme. Sensor nodes located at $CA_{N=3}$ will encrypt all data of every retrieved image, which are transmitted with no coding. Sensor nodes located at $CA_{N=2}$ will apply DWT just once, encrypting only sub-layer $LL_{(1)}$. Finally, sensor nodes located at $CA_{N=1}$ will perform encoding in two DWT levels, encrypting only the sub-layer $LL_{(2)}$. The remaining visual sensors will not encrypt sensed images.

Regarding Scheme 2, Figure 5 presents its generic diagram.

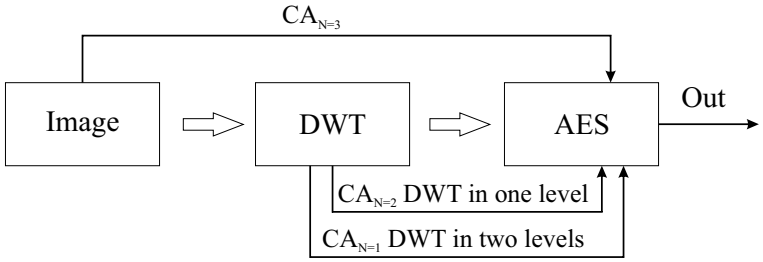


Figure 4. Diagram for Scheme 1.

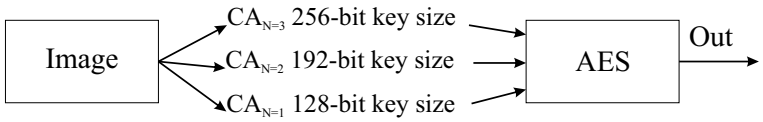


Figure 5. Diagram for Scheme 2.

The main objective of the proposed approach is to save energy when compared with security mechanisms where every sensed data is protected. For that, energy consumption when applying AES encryption in source nodes had to be assessed. Energy consumption was assessed based on definitions in [Potlapally et al. 2006] and parameters as sensed data and key size. The formulation in (3) defines the consumed energy for the performed encryption.

$$EnergyCost_{(Si)} = EKey_{(Si)} + (EByte_{(Si)} * Size_{(Data)}) \quad (3)$$

In equation (3), $EKey_{(Si)}$ represents the energy costs to expand the symmetric key for algorithm Si. The energy consumed per byte in encryption/decryption using the algorithm Si is given by $EByte_{(Si)}$, and $Size_{(Data)}$ is the total data size to be encrypted by the algorithm Si. Using this equation it is possible to estimate energy consumption for any symmetric encryption algorithm.

The values for $EKey_{(Si)}$ and $EByte_{(Si)}$ are described in [Potlapally et al. 2006], for algorithm AES, as presented in Table 1.

Table 1. Energy cost of AES variants [Potlapally et al. 2006].

Key Size (bits)	$EKey_{(AES)}$ (μJ)	$EByte_{(AES)}$ in ECB mode (μJ/B)	$EByte_{(AES)}$ in CBC mode (μJ/B)	$EByte_{(AES)}$ in CFB mode (μJ/B)	$EByte_{(AES)}$ in OFB mode (μJ/B)
128	7.83	1.21	1.62	1.91	1.62
192	7.87	1.42	2.08	2.30	1.83
256	9.92	1.64	2.29	2.31	2.05

We defined some verification scenarios to be considered when assessing energy consumption, as expressed in Table 2.

All performed verifications considered 100 hours of transmissions where all deployed visual sensors periodically transmit images. Moreover, we assumed that each

Table 2. Initial verification scenarios.

Scenario	Security scheme	Image resolution	Key size	$CA_{N=1}$	$CA_{N=2}$	$CA_{N=3}$	Algorithm
1	1	128 x 128 x 8	128 bits	125	125	125	AES / ECB
2	1	256 x 256 x 4	256 bits	125	125	125	AES / ECB
3	2	128 x 128 x 4	vary	90	90	90	AES / ECB
4	2	128 x 128 x 4	vary	90	90	90	AES / CBC

source node collects, encrypts and transmits only one image snapshot per second. Concerning the encryption algorithm, we considered AES in ECB (Electronic Code Book) or CBC (Cipher Block Chaining) modes.

Figure 6 presents the total consumed energy for image encryption in all deployed source nodes, for Scenarios 1, 2, 3 and 4.

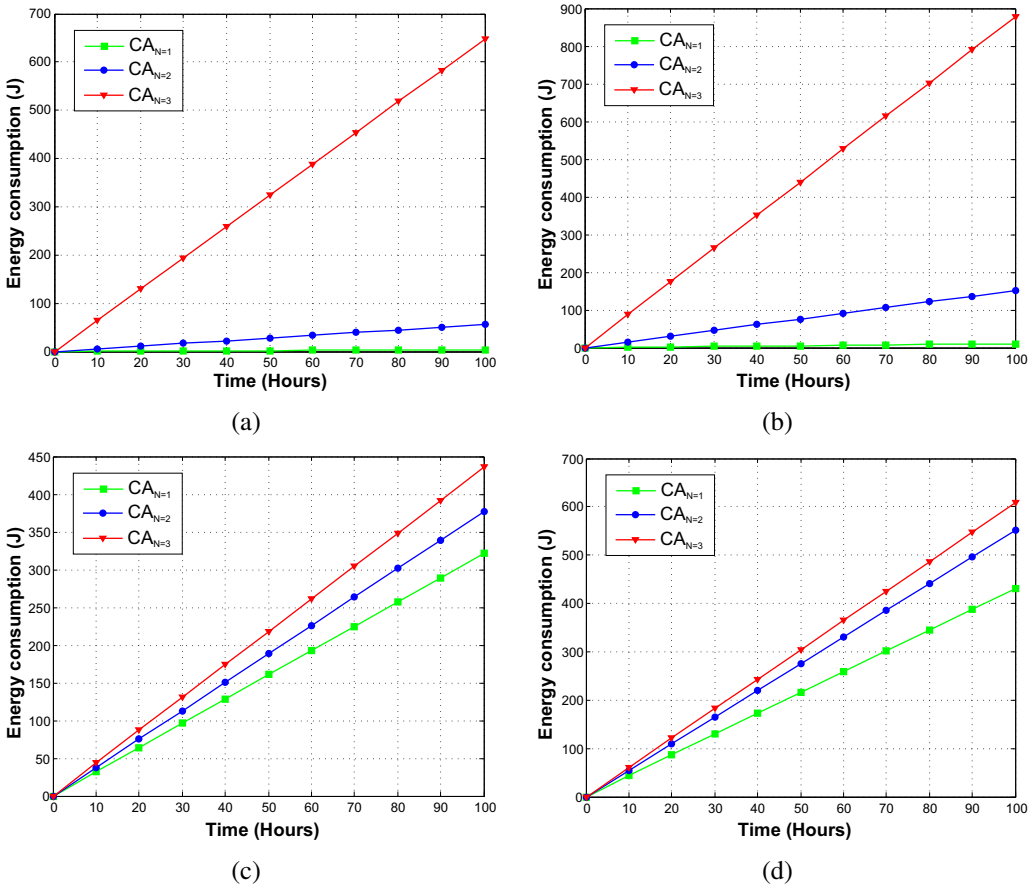


Figure 6. Energy (J) (a) Scenario 1; (b) Scenario 2; (c) Scenario 3; (d) Scenario 4.

As can be seen in Figure 6, energy consumption depends on the adopted security scheme, which directly depends on definitions of confidential areas. In other words, the identification of CAs is central for energy preservation in WVSN, since neither all visual source nodes and viewed areas will have the same security requirements. As expected, energy consumption was higher for confidential areas with higher confidentiality levels.

The second stage of verifications was concerned with energy consumption for an

entire wireless visual sensor network, when the proposed approach is employed. Table 3 presents the configurations for Scenarios 5, 6 and 7, that will be considered for these verifications. For these three scenarios, a total of 270 visual source nodes were deployed, where some of them are associated to $CA_{N=0}$.

Table 3. Additional verification scenarios.

Scenario	Security scheme	Image resolution	Key size	$CA_{N=1}$	$CA_{N=2}$	$CA_{N=3}$	Algorithm
5	1	128 x 128 x 8	128 bits	46	32	67	AES / CBC
6	1	128 x 128 x 8	128 bits	41	56	79	AES / CBC
7	2	128 x 128 x 4	vary	41	56	79	AES / CBC

The presented results in Figure 7 shows the consumed energy for a network ensuring full protection for all sensor nodes (as if all nodes where in a $CA_{N=3}$) and the energy for a network employing the proposed approach. Actually, the energy consumed for CAAP operation is too low compared to the overall data transmissions, with CAAP messages typically sizing less than 20 bytes.

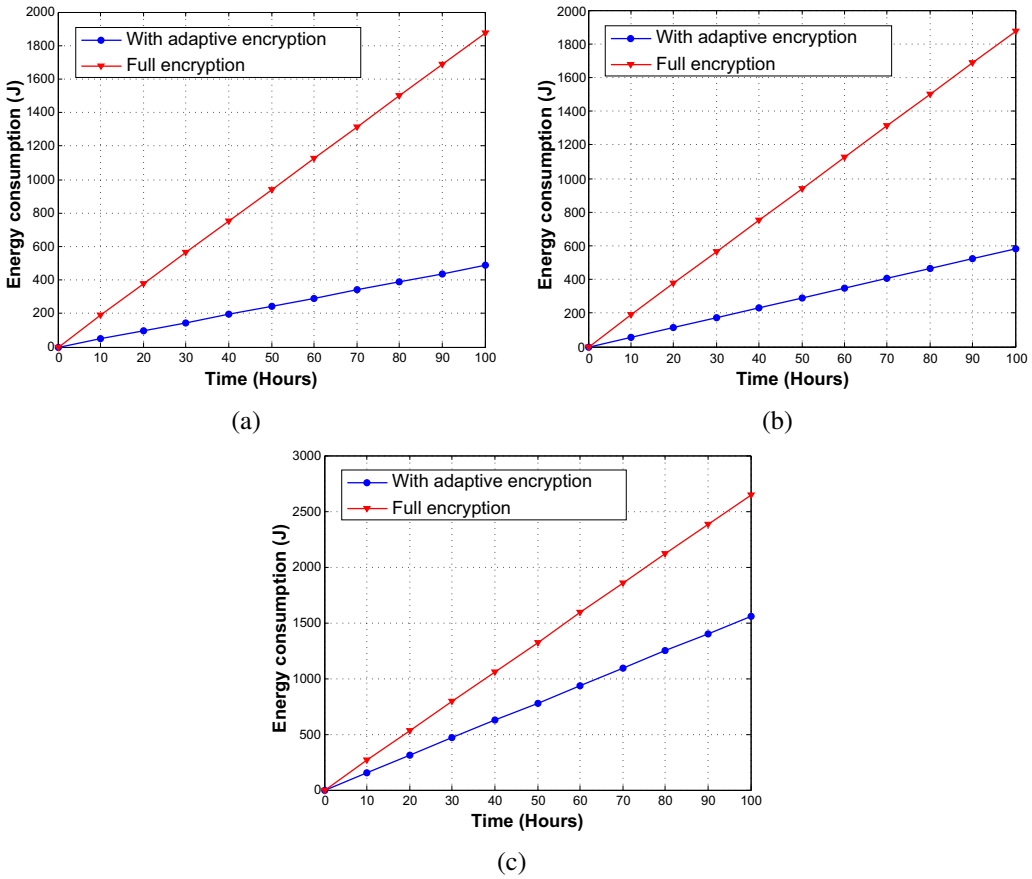


Figure 7. Energy consumption (a) Scenario 5; (b) Scenario 6; (c) Scenario 7.

One can see in Figure 7 that energy consumption in a sensor network without the proposed solution is higher than the consumed energy for the same network when employing the adaptive encryption mechanism. And although it may seem that the ”traditional”

security mechanism is safer, the proposed approach provided high security protection to all critical data.

As a final remark, CAAP may impact the overall latency for assignment of sensor nodes to CAs. Actually, the initial assignment during network startup is not necessarily a concern, but additional CAAP communications will incur in some delay for nodes assignment and use of the proper security scheme. In fact, latency in wireless sensor networks depends on many characteristics, as duty cycling configuration, MAC operation, bit rate, routing protocol, among others. Nevertheless, for any additional delay, and considering that CAAP comprises few messages, the time until new security schemes will be applied due to new CA configurations will be too short comparing to the network lifetime. And such additional delay will not cause a security breach.

After initial verification, we may expect that the proposed solution will bring significant results to wireless visual sensor networks. Actually, fewer energy consumption is achieved while assuring strong protection to sensed data with high confidential requirements.

6. Conclusions

As applications may have different security requirements for monitored areas, this paper proposes an innovative approach that can save energy while assuring acceptable levels of security protection. We defined the concepts and procedures for the adoption of this approach, which was initially designed for wireless visual sensor networks.

This work is not concluded yet. As future works, the proposed approach will be validated in real sensor networks, employing RaspBerry PI sensor motes to implement Secure Schemes and CAAP. Doing so, we plan to validate processing and memory costs of the proposed approach. Moreover, we also want to integrate it with QoS and QoE-based optimizations, achieving secured and prioritized wireless visual sensor networks.

Acknowledgment

The authors would like to acknowledge the support of the Brazilian research agency CNPq (grant number 441459/2014-5), that partially funded this work.

References

Chen, X., Makki, K., Yen, K., and Pissinou, N. (2009). Sensor network security: a survey. *IEEE Communications Surveys & Tutorials*, 11(2):52–73.

Costa, D. G. and Guedes, L. A. (2011). A survey on multimedia-based cross-layer optimization in visual sensor networks. *Sensors*, 11(5):5439–5468.

Costa, D. G., Silva, I., Guedes, L. A., Portugal, P., and Vasques, F. (2014). Selecting redundant nodes when addressing availability in wireless visual sensor networks. In *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*, pages 130–135. IEEE.

Elsabi, E. and Ozdemir, S. (2012). Secure data aggregation in wireless multimedia sensor networks via watermarking. In *Proceedings of the International Conference on Application of Information and Communication Technologies*, pages 1–6.

Gao, R., Wen, Y., Zhao, H., and Meng, Y. (2014). Secure data aggregation in wireless multimedia sensor networks based on similarity matching. *International Journal of Distributed Sensor Networks*, 2014:Article ID 494853, 6 pages.

Gonçalves, D. and Costa, D. G. (2015). A survey of image security in wireless sensor networks. *Journal of Imaging*, 1(1):4–30.

Grangetto, M., Magli, E., and Olmo, G. (2006). Multimedia selective encryption by means of randomized arithmetic coding. *Multimedia, IEEE Transactions on*, 8(5):905–917.

Lecuire, V., Duran-Faundez, C., and Krommenacker, N. (2007). Energy-efficient transmission of wavelet-based images in wireless sensor networks. *Journal on Image and Video Processing*, 2007:1–15.

Nikolakopoulos, G. and Fanakis, N. (2009). A reconfigurable transmission scheme for lossy image transmission over congested wireless sensor networks. In *Proceedings of the International Congress on Image and Signal Processing*.

Nikolakopoulos, G., Kandris, D., and Tzes, A. (2010). Adaptive compression of slowly varying images transmitted over wireless sensor networks. *Sensors*, 10(8):7170–7191.

Ozdemir, S. and Xiao, Y. (2009). Secure data aggregation in wireless sensor networks: A comprehensive overview. *Computer Networks*, 53(12):2022–2037.

Pescaru, D. and Curiac, D.-I. (2014). Anchor node localization for wireless sensor networks using video and compass information fusion. *Sensors*, 14(3):4211–4224.

Potlapally, N. R., Ravi, S., Raghunathan, A., and Jha, N. K. (2006). A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *Mobile Computing, IEEE Transactions on*, 5(2):128–143.

Sen, J. (2009). A survey on wireless sensor network security. *International Journal of Communication Networks & Information Security*, 1(2):59–82.

Simplicio, M. A., Oliveira, B. T., Margi, C. B., Barreto, P. S. L. M., Carvalho, T. C. M. B., and NÃslund, M. (2013). Survey and comparison of message authentication solutions on wireless sensor networks. *Ad Hoc Networks*, 11:1221–1236.

Wang, Q.-X., Xu, T., and zhou Wu, P. (2011). Application research of the aes encryption algorithm on the engine anti-theft system. In *Proceedings of IEEE International Conference on Vehicular Electronics and Safety*, pages 25–29.

Wang, W., Hempel, M., Peng, D., Wang, H., Sharif, H., and Chen, H.-H. (2010). On energy efficient encryption for video streaming in wireless sensor networks. *IEEE Transactions on Multimedia*, 12(5):417–426.

Wang, Y., Attebury, G., and Ramamurthy, B. (2013). Security issues in wireless sensor networks: a survey. *International Journal of Future Generation Communication and Networking*, 6(5):97–116.

Winkler, T. and Rinner, B. (2014). Security and privacy protection in visual sensor networks: A survey. *ACM Computing Surveys*, 47(1):97–116.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer networks*, 52(12):2292–2330.

Trilha Principal do SBRC 2016
Sessão Técnica 21
Redes LTE e Rádio

Um mecanismo baseado em SDN para flexibilizar o controle de tráfego em redes LTE

Luciano Jerez Chaves[1,2], **Islene Calciolari Garcia**[2], **Edmundo R. Mauro Madeira**[2]

[1] Depto. de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF)
Rua José Lourenço Kelmer, s/n. São Pedro, Juiz de Fora – MG.

[2] Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251. Cidade Universitária, Campinas – SP.

{lchaves, islene, edmundo}@ic.unicamp.br

Abstract. *The rapid growth in mobile Internet access is forcing operators to handle network traffic more resourcefully, and the Software Defined Networking (SDN) has emerged as a feasible solution to this problem. SDN is pointed as an enabling technology to overcome current limitations, reducing costs, increasing network scalability, and making possible upcoming 5G networks. This paper presents a SDN-based mechanism for traffic admission and routing in Long-Term Evolution (LTE) core networks. Simulation results underline the benefits that can be achieved with this mechanism. It also discusses how this and other literature proposals can contribute to make these networks more flexible.*

Resumo. *O rápido crescimento no acesso móvel à Internet está forçando as operadoras a gerenciar melhor o tráfego na rede, e as Redes Definidas por Software (SDN) surgiram como uma possível solução para este problema. O SDN é apontado como uma tecnologia facilitadora para superar as limitações atuais, reduzindo os custos, aumentando a escalabilidade e viabilizando as iminentes redes 5G. Este artigo apresenta um mecanismo baseado em SDN para controle de admissão e roteamento de tráfego no núcleo das redes Long-Term Evolution (LTE). Resultados de simulações destacam os benefícios que podem ser alcançados com este mecanismo. Também é discutido como esta e outras propostas na literatura contribuem para a flexibilização dessas redes.*

1. Introdução

As redes de banda larga móvel estão registrando um crescimento significativo de tráfego nos últimos anos. De acordo com a [Cisco Systems, Inc. 2015], o tráfego móvel global cresceu 69% em 2014, e é esperado um aumento de 10 vezes até 2019. Não apenas o tráfego, mas também o número de dispositivos conectados está crescendo. Atualmente, a tecnologia 4G contribui com apenas 2,9% das conexões móveis, mas já produz aproximadamente 30% de todo o tráfego. Além disso, nos próximos 5 anos é esperado um crescimento anual de 46% no número de dispositivos conectados, o que impõe desafios significantes para as redes atuais e futuras. A crescente demanda levou ao aumento da capacidade de acesso através do reuso da frequência com células menores, enquanto as iminentes redes 5G devem convergir para redes heterogêneas, com diferentes tecnologias sem fio. Este cenário está resultando numa grande quantidade de estações base que necessitam de uma infraestrutura com alta conectividade. Desta forma, é preciso simplificar e otimizar essas redes de núcleo.

Neste contexto surgiram novos paradigmas como a Virtualização das Funções de Rede (NFV) [ETSI NFV 2012], que visa utilizar as tecnologias de computação em nuvem para consolidar as funções de rede tradicionalmente implementadas em *hardware* dedicado. Além disso, as Redes Definidas por *Software* (SDN) e o protocolo OpenFlow [ONF 2012] surgiram como alternativas capazes de proporcionar uma mudança na direção de um modelo centrado no fluxo, que utiliza *hardware* programável de baixo custo, um controlador logicamente centralizado, e aplicações em *software* que usam as informações expostas pelo controlador para orquestrar os serviços da rede. O SDN pode ser visto como um facilitador para a implantação do NFV, e juntos podem simplificar as redes de telefonia celular e reduzir os custos de gerência, construindo o conceito de Redes Móveis Definidas por *Software* (SDMN).

Diferentes trabalhos na literatura discutem como as redes *Long-Term Evolution* (LTE) existentes podem se beneficiar da tecnologia SDN, trazendo agilidade e flexibilidade para as operadoras que oferecem este serviço 4G. Em [Li et al. 2012], os autores abordam as principais limitações das redes atuais, como equipamentos caros, centralização das funções de rede e soluções proprietárias vendidas pelos fabricantes. Eles também mostram como o SDN pode substituir completamente os elementos da arquitetura do núcleo LTE, resultando em um plano de dados simplificado, com todo o controle implementado em *software*. O trabalho de [Hampel et al. 2013] também argumenta em favor do SDN em conjunto com as redes móveis, dedicando atenção especial na gerência de túneis no núcleo da rede, estratégia comum nas arquiteturas de telefonia celular. Apesar do SDN ter sido concebido para ambientes cabeados, não apenas o núcleo da rede pode se beneficiar com ele. No trabalho de [Arslan et al. 2015], os autores argumentam que as redes sem fio já oferecem altas taxa de transferência, mas sofrem de coordenação eficiente entre as estações base. Eles investigam como os princípios de SDN também podem ser aplicados no contexto das redes de acesso sem fio, e concluem que desacoplar o plano de dados do plano de controle facilita o controle de interferência entre células vizinhas. Em relação ao futuro das redes móveis, os trabalhos de [Wang et al. 2015], [Kyung et al. 2015] defendem que um diferencial dos sistemas 5G será como orquestrar todo o controle das redes, e o SDN é a escolha natural.

Neste artigo é discutido como a tecnologia SDN, especificamente o protocolo OpenFlow, pode ser usado para o controle de admissão e roteamento de tráfego em redes LTE. Como uma das contribuições, é proposto um mecanismo capaz de gerenciar a reserva de recursos no núcleo da rede, explorando as funcionalidades do SDN para o controle de admissão e roteamento de tráfego, e tornando mais justa a taxa de bloqueio entre os usuários. O simulador de redes *ns-3* foi utilizado para validar a proposta, fazendo uso de um novo módulo OpenFlow versão 1.3 que foi apresentado pelos mesmos autores deste trabalho em [Chaves et al. 2016]. Além disto, este artigo também lista trabalhos recentes envolvendo a integração entre OpenFlow e LTE, destacando diferentes contribuições que exploram a flexibilidade oferecidos pelo SDN.

O restante deste documento está organizado como segue: a Seção 2 apresenta os conceitos básicos em SDN e LTE, enquanto a Seção 3 apresenta o mecanismo para controle de admissão e roteamento de tráfego proposto neste trabalho, incluindo a avaliação de desempenho através de simulações. Por fim, outros trabalhos na literatura são apresentados na Seção 4, seguidos pelas conclusões na Seção 5.

2. Arquiteturas SDN e LTE

2.1. Redes Definidas por *Software* (SDN)

O SDN é um novo paradigma que foi projetado para tornar as redes mais ágeis, simples e com baixo custo. Na arquitetura SDN, os planos de controle e de dados são desassociados: a inteligência da rede é logicamente centralizada, e a infra-estrutura da rede é abstraída das aplicações. Ao centralizar a inteligência, fica mais fácil tomar decisões com base em uma visão global da rede, ao contrário do que acontece atualmente no modelo em camadas. A Figura 1 retrata a visão lógica dessa arquitetura, segundo a *Open Networking Foundation* (ONF) [ONF 2012].

A inteligência da rede está localizada na camada de controle, implementada em *software*. Em direção à camada de aplicação, a arquitetura suporta um conjunto de APIs para implementar serviços comuns de rede, como roteamento e controle de admissão. Na direção contrária, o protocolo OpenFlow [McKeown et al. 2008] define a comunicação entre os controladores e os *switches* físicos da rede.

O protocolo OpenFlow é padronizado pela ONF, e especifica as primitivas básicas que podem ser usadas pelo controlador para programar o plano de dados dos *switches*, usando o conceito de fluxos para identificar os tráfegos na rede. O plano de dados de um *switch* OpenFlow consiste em uma ou mais tabelas de fluxo, uma tabela de grupo e uma tabela de *meter*. Cada entrada na tabela de fluxo possui campos usados para identificar os fluxos e um conjunto de instruções para aplicar aos pacotes correspondentes. Se uma entrada correspondente for encontrada para um pacote, as instruções são aplicadas e podem conter ações que modifiquem o pacote, determinem o encaminhamento, o processamento em grupo e/ou a aplicação de *meters*. Uma ação de saída encaminha o pacote para uma porta de saída do *switch*, enquanto uma ação de grupo executa conjuntos de ações com semântica de encaminhamento complexa. Por fim, as *meters* são usadas para medir a taxa de transmissão do fluxo, e permitem limitar a vazão do mesmo em um determinado valor. O *switch* também pode enviar os pacotes para o controlador quando não são encontradas entradas para o mesmo nas tabelas de fluxo, ou simplesmente descartá-lo. O leitor pode consultar [OpenFlow 1.5.1 2015] para obter mais detalhes sobre o protocolo.

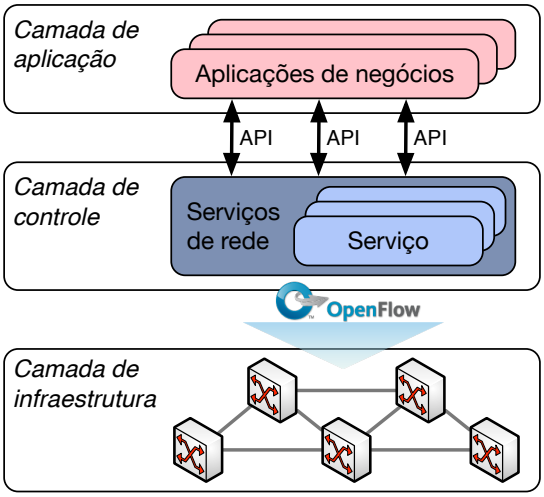


Figura 1. A arquitetura SDN segundo a ONF.

2.2. *Long-Term Evolution* (LTE)

O LTE é uma coleção de padrões para redes móveis de alta velocidade, composto pela rede de acesso sem fio *Evolved Universal Terrestrial Radio Access Network* (E-UTRAN) e pelo núcleo *Evolved Packet Core* (EPC). Juntos, eles formam o *Evolved Packet System* (EPS), como ilustrado na Figura 2. No E-UTRAN, o *Enhanced Node B* (eNB) é o elemento central, que fornece a interface sem fio para transmissão de pacotes de dados e de controle entre o núcleo e o usuário (UE). Já o EPC é um arquitetura IP composta pelos *gateways* de serviço (S-GW) e de pacotes (P-GW), pelo *Mobility Management Entity* (MME), pelo *Home Subscriber Server* (HSS) e pelo *Policy Control and Charging Rules Function* (PCRF). Os *gateways* são responsáveis pelo encaminhamento de pacotes no plano de dados. O P-GW fornece conectividade entre os UEs e a Internet, controlando alocação de endereços IP e realizando a classificação de tráfego. O S-GW lida com os túneis, mobilidade, e também atua como âncora durante o processo de *handover*. O MME é o elemento-chave no plano de controle, responsável pela gerência de mobilidade do UE, *roaming*, segurança e autenticação. O HSS armazena informações relativas aos UEs e suas assinaturas, enquanto o PCRF lida com as políticas de serviço.

O EPS utiliza o conceito de *bearers* para rotear o tráfego IP. Um *bearer* está associado aos fluxos de pacotes IP que recebem um mesmo tratamento de Qualidade de Serviço (QoS) entre o UE e o P-GW. Existem nove identificadores de classes de QoS (QCI) definidos pelo LTE, que especificam atributos como prioridade de escalonamento, taxa de perda e atraso médio [3GPP 23.203 2015]. Assim, os serviços que demandam por QoS específico podem ser mapeados para os *bearers* adequados com base no QCI.

Um *bearer* pode ser classificado em *bearer* com taxa de dados mínima garantida (GBR) para os qual é alocado banda para transmissão dedicada, ou em *bearer* sem garantia de taxa de dados (Non-GBR). Todo UE possui um *bearer* Non-GBR padrão, e pode ou não ter outros *bearers* dedicados, conforme a demanda dos tráfegos. Como um *bearer* passa por diversas interfaces dentro do EPS, ele precisa ser mapeado e identificado nas camadas inferiores da arquitetura de rede. Especificamente pelas interfaces S1 e S5/S8, túneis GTP (sobre UDP / IP) são utilizados para encapsular e rotear os pacotes de um mesmo *bearer*. Neste caso, os endereços IP de origem e destino do túnel e o ID do túnel (TEID) são usados para identificá-lo na rede.

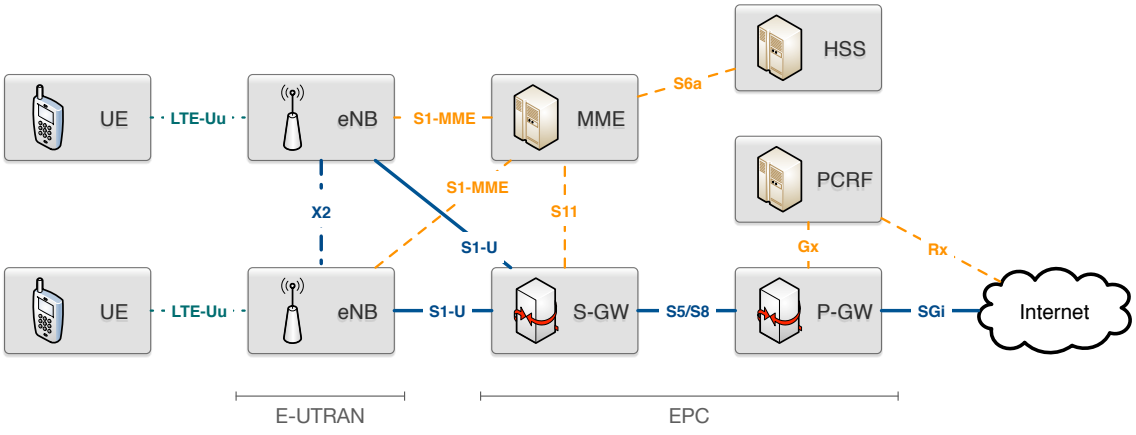


Figura 2. A arquitetura EPS.

3. Controle de tráfego em redes OpenFlow + LTE

3.1. Integração entre OpenFlow e LTE

A proposta de integração entre o protocolo OpenFlow e as redes LTE neste trabalho envolve o uso de *switches* OpenFlow para construir a infraestrutura de *backhaul* da rede, conectando os eNBs aos *gateways* no núcleo EPC. Essa infraestrutura de rede é responsável por transportar todo o tráfego das interfaces S1 e X2, que é encapsulado dentro dos protocolos GTP / UDP / IP. Considerando o roteamento de tráfego e tendo em vista a compatibilidade com as redes LTE convencionais, esta proposta de integração preserva o encapsulamento do tráfego, de maneira que nenhuma modificação nos padrões é necessária e nenhum elemento da arquitetura EPS que manipula as interfaces S1 e X2 precisa ser substituído. Em contraste com outras soluções na literatura, manter a compatibilidade com os túneis GTP permite a fácil integração dos novos elementos OpenFlow em infraestruturas já existentes, apenas com algum ajuste no *software* do controlador.

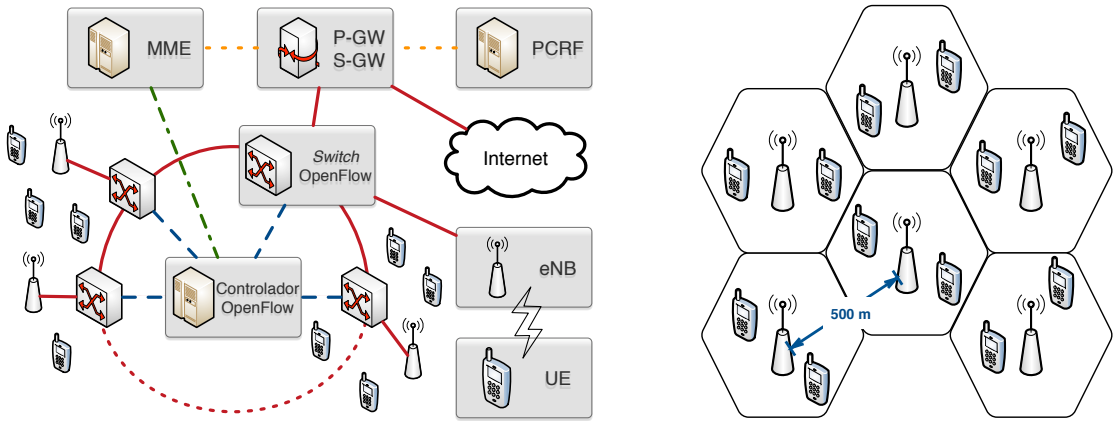
Considerando a manutenção do protocolo GTP, extensões são necessárias para que o OpenFlow possa lidar com os túneis. Duas abordagens foram consideradas:

1. Incrementar o *switch* OpenFlow com portas virtuais, capazes de remover os cabeçalhos do túnel quando o pacote entra no *switch*, e reconstruir o pacote quando o mesmo deixa o equipamento. Essa abordagem permite regras de fluxo com ajuste fino, porém resulta em maior demanda por processamento para lidar com a manipulação dos cabeçalhos.
2. Estender tanto o *switch* como o protocolo OpenFlow com novos campos para o cabeçalho GTP, em uma abordagem mais simples onde não há operações de des/encapsulamento. Porém, somente os campos dos cabeçalhos dos túnel (os cabeçalhos IP e UDP mais externos) podem ser usados nas regras de fluxo.

Para este trabalho foi escolhida a segunda abordagem, considerando que os cabeçalhos do túnel já contêm as informações necessárias para os processos de controle de admissão e roteamento. Dois novos campos OpenFlow para o protocolo GTP foram especificados, seguindo a estratégia presente em [Kempf et al. 2012]: um campo de *flags* GTP, com dois *bytes*; e um campo de TEID, com quatro *bytes*.

Em relação à topologia da rede de infraestrutura, a Figura 3a mostra a porção cabeada do cenário proposto. A topologia em anel possui um número arbitrário de *switches* OpenFlow. O *gateway* unificado S-GW / P-GW[1] é conectado ao primeiro *switch*, enquanto os outros *switches* estão conectados aos eNBs. As conexões entre os *switches* OpenFlow são feitas através de enlaces Ethernet *full-duplex*, com taxa de transmissão configurável e atraso médio estimado em $100\,\mu$s, valor típico para fibra metropolitana de aproximadamente 20 Km. A infraestrutura de rede foi construída sobre tecnologia Ethernet pois esta é considerada a mais efetiva para o transporte de pacotes IP, além de ter custo reduzido que facilita a escalabilidade da rede [Howard 2011]. A topologia em anel foi escolhida pois muitas das redes existentes já são configuradas em anel. Além disso, estudos afirmam que o anel é a topologia mais eficiente em termos de proteção com custo reduzido [Nadiv e Naveh 2010]. De qualquer maneira, é possível considerar outras topologias substituindo as conexões entre os *switches*.

[1]O *gateway* unificado é necessário por conta da limitação de implementação no simulador *ns-3*.



(a) Rede de infraestrutura cabeada. (b) Rede de acesso sem fio.

Figura 3. Topologia para o cenário de integração entre OpenFlow e LTE.

A Figura 3b mostra a topologia da rede de acesso sem fio, em que cada eNB cobre um número arbitrário de UEs ativos. Os eNBs são posicionados em uma estrutura hexagonal, a 30 m da superfície, e com uma distância de 500 m entre as antenas. Os UEs conectados à cada eNB estão distribuídos ao redor da antena, a 1,5 m da superfície.

Para gerenciar os *switches* OpenFlow é preciso um controlador capaz de interagir com o elemento MME da arquitetura EPS, para que o controlador fique ciente dos procedimentos de ativação, modificação e desativação de um *bearer*. Essa interação entre o controlador e o MME é destacada na Figura 4, que traz a sequência de mensagens do processo de ativação de um *bearer* dedicado. O MME, ao receber uma solicitação de criação de *bearer*, envia uma requisição ao controlador, que irá realizar as funções de controle de admissão e roteamento, retornado ao MME para que o processo continue. Os procedimentos para modificação e desativação de *bearers* seguem a mesma lógica.

Neste trabalho, as funções de responsabilidade do controlador OpenFlow incluem roteamento de tráfego pela rede de infraestrutura e o controle de admissão de tráfegos com reserva de recursos (*bearers* GBR). Essas funcionalidades são detalhadas a seguir.

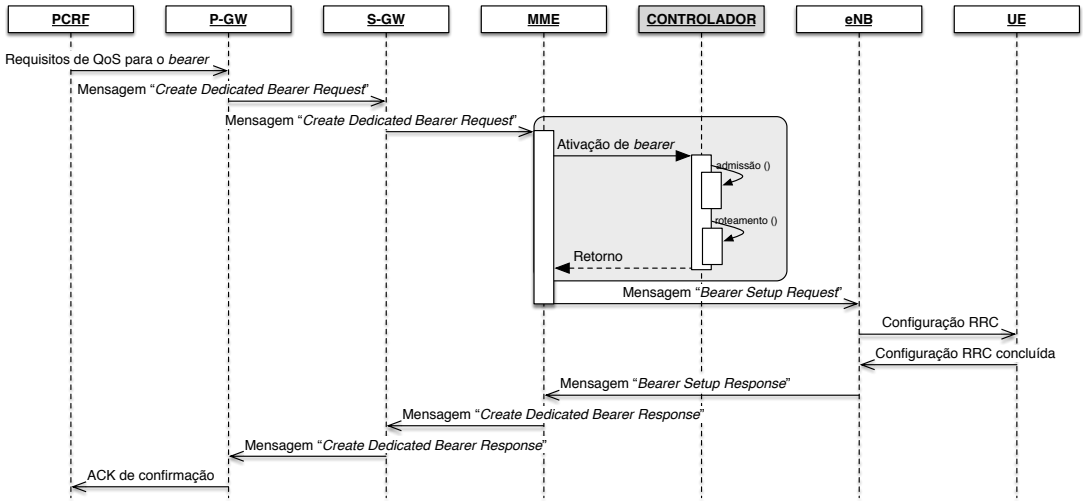


Figura 4. Integração entre o controlador OpenFlow e o MME.

3.2. Roteamento de tráfego pelo OpenFlow

O roteamento é o processo de selecionar caminhos em uma rede, e uma das grandes vantagens oferecidas pelo paradigma SDN é a possibilidade de explorar diferentes estratégias de roteamento graças ao controle centralizado em *software*. O controlador mantém uma visão global da rede, tanto em termos de topologia como em largura de banda disponível em cada enlace. Com esta informação, é possível rotear o tráfego através da rede usando abordagens clássicas, como o menor caminho entre os elementos de origem e destino. Contudo, algoritmos mais complexos podem ser empregados para rotear o tráfego, baseado na largura de banda solicitada e nas especificações de QoS de cada tráfego. Até mesmo sofisticadas estratégias de engenharia de tráfego podem ser utilizadas neste cenário, como modelar o problema de alocação dos fluxos na rede como um sistema linear e utilizar ferramentas de *software* para encontrar uma solução ótima.

Para o cenário de integração proposto neste trabalho, quando um procedimento de ativação de *bearer* é iniciado no EPS, o controlador é notificado pelo MME e procura por uma rota aceitável, instalando as regras de roteamento nos *switches* antes mesmo da conclusão da criação do *bearer* e do início do tráfego. Como o tráfego é encapsulado usando o GTP, os campos de cabeçalhos expostos ao OpenFlow são aqueles referentes ao túnel. Para permitir o roteamento dos fluxos baseado no túnel, o *switch* considera apenas o identificador TEID do túnel GTP para rotear o tráfego. Essa abordagem simplifica as regras de fluxo instaladas nos *switches*, o que acelera a busca no *hardware*. As regras de fluxo instaladas são configuradas para desinstalação automática quando não há fluxo de pacotes por tempo predeterminado. O controlador também pode remover as regras de fluxo quando o procedimento de desativação de *bearer* é notificado pelo MME.

Para a infraestrutura de rede em anel deste trabalho, as opções de rota entre quaisquer elementos são duas: direção horária ou anti-horária. Desta forma, o controlador OpenFlow foi programado para trabalhar com duas políticas de roteamento distintas:

- A política do *Menor Caminho Sempre*, onde o tráfego é sempre roteado na direção de menor caminho entre os elementos de origem e destino;
- A política do *Menor Caminho Primeiro*, que tenta rotear o tráfego pelo menor caminho quando possível, senão envia o tráfego na direção oposta quando necessário. Esta abordagem explora a flexibilidade oferecida pelo OpenFlow na seleção de rotas independentes para tráfegos com mesma origem e destino.

3.3. Controle de admissão de *bearers* GBR

Toda vez que um novo *bearer* é ativado, modificado ou desativado na rede LTE, o controlador precisa estar ciente para instalar, atualizar ou remover as regras de fluxo usadas no roteamento. Além do roteamento, o controlador também é responsável por realizar o controle de admissão de *bearers* do tipo GBR, que demandam pela alocação de largura de banda na infraestrutura de rede para garantir as especificações do QCI associado. Desta forma, o controlador precisa primeiro verificar se existe banda suficiente em todos os enlaces da rota selecionada. Quando a largura de banda solicitada pode ser atendida, o procedimento de ativação/modificação do *bearer* prossegue normalmente. Caso contrário, se a política de roteamento permitir, o controlador procura por um novo caminho que possa atender à demanda solicitada. Se não houver uma rota aceitável, a requisição é bloqueada. Quando isto acontece, o MME é notificado e a rede aborta o processo de ativação/modificação do *bearer*.

Observe que o mecanismo de admissão deste trabalho é apenas para o controle de *bearers* GBR. Uma vez que um UE tenha um *bearer* GBR bloqueado por falta de recursos disponíveis, ele tem a opção de solicitar a criação de um novo *bearer* Non-GBR para o tráfego, ou mesmo roteá-lo através do *bearer* padrão (sem nenhuma garantia de QoS inicialmente desejada). Este controle de admissão é complementar à outros mecanismos que possam existir na rede de acesso sem fio como, por exemplo, a admissão de usuários pela estação base para efeitos de escalonamento de recursos no eNB.

3.4. Simulações e resultados

O simulador de redes *ns-3* foi utilizado para validar os mecanismos propostos neste trabalho. O *ns-3* possui um módulo LTE que implementa o plano de dados da arquitetura EPS. A Tabela 1 lista os parâmetros que foram ajustados no simulador para a rede LTE. Os demais parâmetros não listados tiveram seu valor padrão preservados.

Para construir o cenário que integra o LTE ao OpenFlow, foi utilizado também o módulo `OFSwitch13`, que implementa o OpenFlow versão 1.3 [Chaves et al. 2016]. O `OFSwitch13` é uma alternativa ao desatualizado módulo OpenFlow disponível na distribuição do *ns-3*. Ele oferece o plano de dados do *switch*, o canal de comunicação com o controlador, e uma interface para a construção de controladores especializados.

Na rede de infraestrutura cabeada apresentada na Figura 3a, as conexões no anel foram configuradas em 100 Mbps, e foram avaliados cenários com 5 e com 11 *switches* OpenFlow (4 e 10 eNBs, respectivamente). O primeiro *switch* é sempre conectado ao *gateway* unificado, e existe um eNB conectado a cada um dos demais *switches*, que estão numerados em sentido horário. Foram considerados um cenário com carga igualmente distribuída entre todos os eNBs (cenário balanceado), e um cenário em que 30% dos UEs estão conectados aos eNBs da primeira metade do anel, enquanto os 70% restantes estão distribuídos entre os eNBs da segunda metade (cenário desbalanceado). Esta configuração desbalanceada de carga foi intencionalmente escolhida para destacar os benefícios que podem ser alcançados com a escolha seletiva de rotas oferecida pelo OpenFlow. Nas simulações, o mecanismo de controle de admissão foi configurado para reservar no máximo 40% da capacidade dos enlaces para os tráfegos sobre *bearers* GBR.

Como nas redes LTE as chamadas de voz são realizadas por VoIP, e grande parte do tráfego de pacotes é para acesso a conteúdo multimídia, foram consideradas duas aplicações cliente/servidor para gerar o tráfego das simulações:

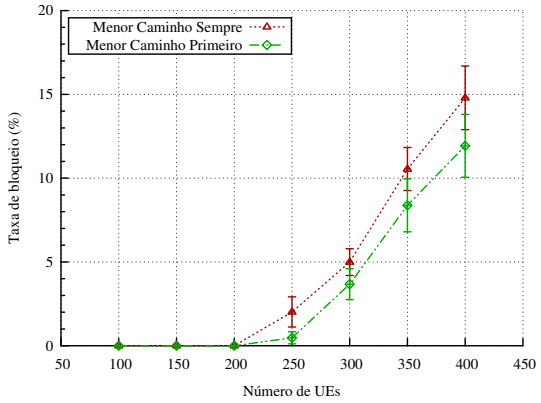
Tabela 1. Parâmetros de simulação do LTE no *ns-3*.

Parâmetro	Valor
Frequência central	2100 MHz
Largura de banda do canal	20.0 MHz
Potência de transmissão do eNB	46 dBm
Potência de transmissão do UE	18 dBm
Periodicidade do SRS	320 ms
Modelo de propagação	`OhBuildingsPropagationLossModel`
Escalonador MAC	`CqaFfMacScheduler`

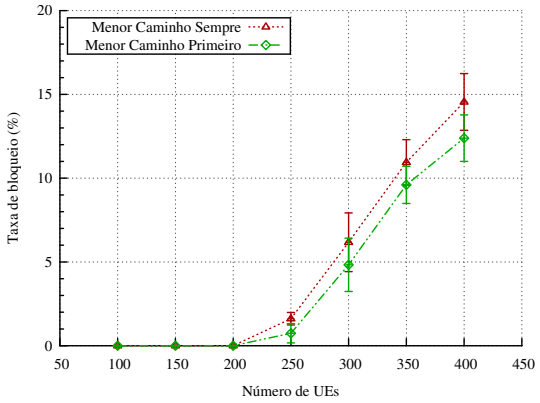
- *Aplicação VoIP*, com tráfego bidirecional sobre protocolo UDP, codificado com o *codec* G.729 (≈8,0 Kbps) e mapeado para o *bearer* GBR com QCI 1 (adequado para tráfegos de voz). O tempo médio de uma ligação é esperado em 100 segundos, e é definido por um gerador normal $\mathcal{N}(100, 100)$.
- *Streaming de vídeo*, com tráfego *downlink* sobre protocolo UDP, gerado de acordo com *traces* de vídeo MPEG-4 [Fitzek e Reisslein 2001] e mapeado para o *bearer* GBR com QCI 2 (adequado para tráfego de vídeo em tempo real). A taxa de transmissão média varia entre 100 Kbps e 1100 Kbps, a depender do *trace* que é aleatoriamente selecionado. A duração média do vídeo é esperada em 90 segundos, e é definida por um gerador normal $\mathcal{N}(90, 225)$.

As duas aplicações geram tráfegos com diferentes requisitos de QoS na rede. Um *bearer* de cada tipo pode ser solicitado por cada UE, que gera requisições seguindo um *processo de Poisson* com taxa $\lambda = 0.\bar{3}$ requisições por minuto. Assim, quanto mais UEs ativos na rede, maior é a taxa agregada do processo de Poisson associada à carga na rede.

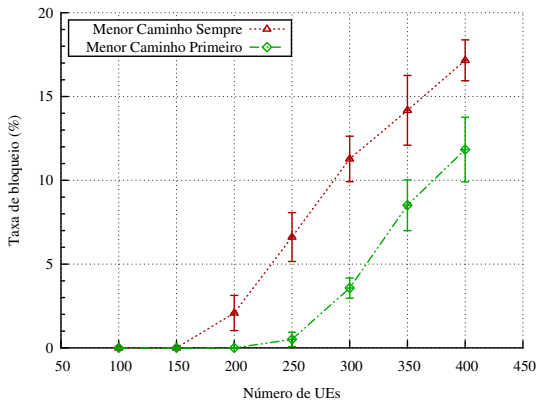
A Figura 5 apresenta os resultados de simulações com intervalo de confiança de 95%. Os gráficos mostram a porcentagem de bloqueio para as requisições de *bearers* GBR feitas pelo mecanismo de controle de admissão, e compara os resultados entre as duas políticas disponíveis no mecanismo de roteamento: a política de *menor caminho sempre* e a política de *menor caminho primeiro*. São apresentados os resultados para as



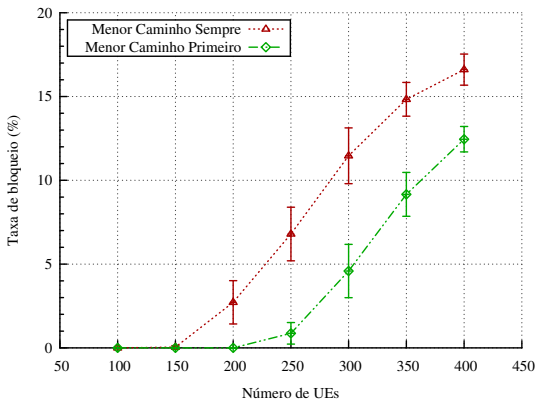
(a) 4 eNBs e carga balanceada.



(b) 10 eNBs e carga balanceada.



(c) 4 eNBs e carga desbalanceada.

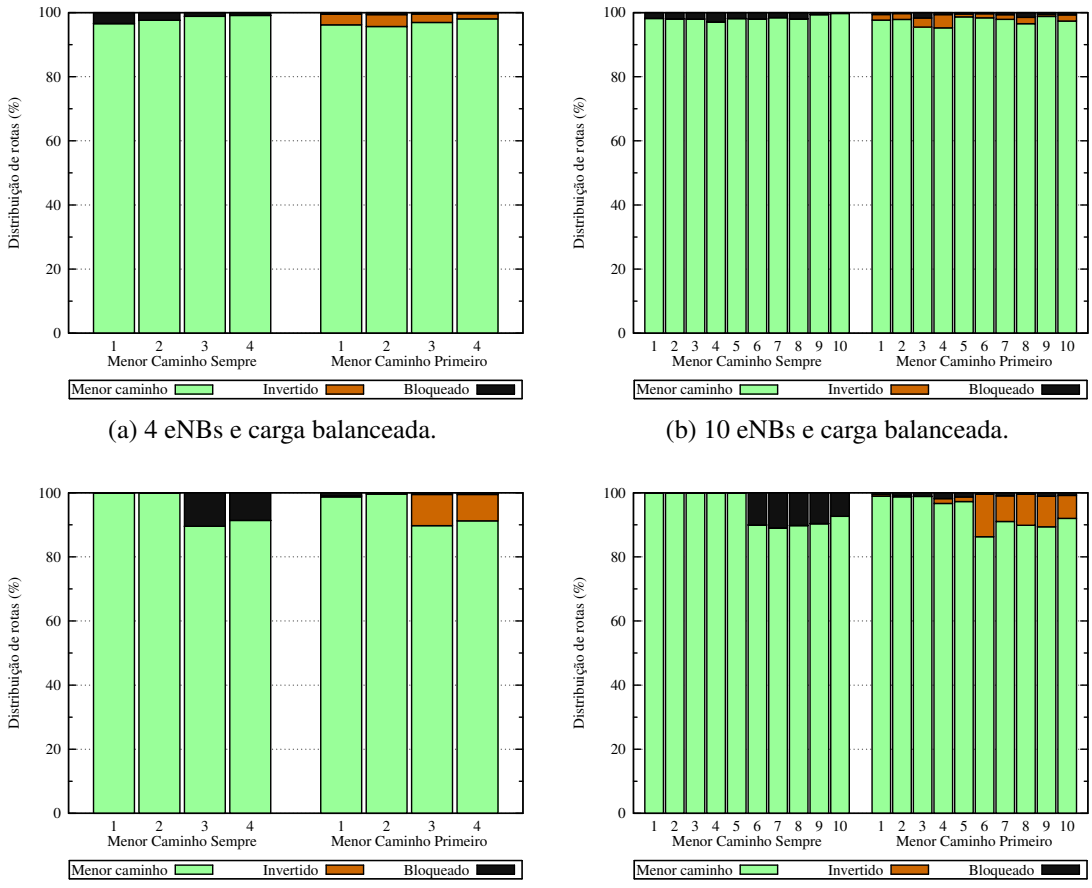


(d) 10 eNBs e carga desbalanceada.

Figura 5. Taxa de bloqueio para requisições de *bearers* GBR.

configurações de cenário variando o tamanho do anel e a distribuição de carga, com diferentes números de usuários. As Figuras 5a e 5b mostram a taxa de bloqueio nos cenários com a carga balanceada, onde os UEs estão igualmente distribuídos entre todos os eNBs. Não há diferenças significativas nos resultados entre os cenários com 4 e 10 eNBs. Nos dois casos é possível perceber que a demanda por reserva de banda na rede para *bearers* GBR não pode mais ser totalmente atendida a partir de 200 UEs, e os bloqueios ocorrem para as duas estratégias de roteamento. Na média, a política de *menor caminho primeiro* diminuiu a taxa de bloqueio, mas não há ganhos estatisticamente significantes dentro do intervalo de confiança. Já as Figuras 5c e 5d mostram a taxa de bloqueio para o cenário com carga desbalanceada. Novamente, não há diferenças significativas nos resultados entre os cenários com 4 e 10 eNBs, porém é fácil perceber que a política de roteamento de *menor caminho primeiro* reduz significativamente a taxa de bloqueio nestes cenários, diminuindo em cerca de 3,7 pontos percentuais o bloqueio médio e permitindo um aumento próximo de 33% no número de UEs antes de iniciar bloqueios.

Para justificar a redução na taxa de bloqueio, a Figura 6 mostra a proporção de tráfegos bloqueados, roteados pelo menor caminho ou pelo caminho inverso, para cada um dos eNBs nas simulações com 250 UEs. Seguindo a mesma ordem, as Figuras 6a e 6b mostram essa distribuição para os cenários com carga balanceada. Observe que para a política de *menor caminho sempre* há uma pequena quantidade de bloqueios aconte-



(a) 4 eNBs e carga balanceada.

(b) 10 eNBs e carga balanceada.

(c) 4 eNBs e carga desbalanceada.

(d) 10 eNBs e carga desbalanceada.

Figura 6. Escolha de rotas para requisições de *bearers* GBR.

cendo em todos os eNBs, enquanto que para a política de *menor caminho primeiro* existem também tráfegos roteados pelo caminho contrário (invertido). Estes poucos tráfegos exploram alguma largura de banda disponível no caminho inverso em determinados momentos, e contribuem para a pequena redução na taxa de bloqueio. Já para os cenários com carga desbalanceada, as Figuras 6c e 6d confirmam os benefícios alcançados com a flexibilidade na escolha das rotas. Enquanto a primeira metade do anel possui poucos usuários e, consequentemente, pouca demanda por tráfego, os enlaces entre os *switches* na segunda metade não conseguem atender todas as requisições. Assim, a política de roteamento de *menor caminho sempre* gera um desequilíbrio na taxa de bloqueio, sendo os UEs do lado sobrecarregado da rede mais prejudicados. Com a política de *menor caminho primeiro*, é possível redirecionar a demanda excedente do lado sobrecarregado para o outro lado do anel, diminuindo e equilibrando a taxa de bloqueio entre todos os UEs.

4. Contribuições para as redes SDMN flexíveis

4.1. Trabalhos relacionados na literatura

No contexto de redes SDMN, existem diversos trabalhos na literatura que abordam a integração entre SDN e LTE. A Tabela 2 resume as principais contribuições recentes nesta área. Alguns trabalhos são motivacionais, discutem as limitações das redes existentes e como as redes SDMN podem solucionar estes problemas. Outros autores defendem que o conceito de SDN também pode ser aplicado em redes de acesso sem fio, e há trabalhos nesta direção. Existe também um grupo de propostas que são focadas em novas arquiteturas SDN para permitir que as futuras redes móveis sejam mais robustas do que as atuais. Em linhas gerais, as novas arquiteturas substituem o plano de dados e centralizam o controle em *software*, seguindo a direção das redes 5G. Apesar das propostas serem escaláveis, um dos desafios está na integração das novas arquiteturas com as redes legadas, pois a compatibilidade com o padrão atual nem sempre é preservada.

Por outro lado, há soluções que se concentram em desafios específicos. Dentre os tópicos abordados estão a gerência de túneis nas arquiteturas móveis, o controle de congestionamento no núcleo da rede, a gerência distribuída de mobilidade dos usuários, mecanismos para balanceamento de carga na rede de infraestrutura, e a resiliência para falhas de equipamentos e enlaces.

Tabela 2. Diferentes contribuições para redes SDMN flexíveis.

Referência	Contribuição
[Chaves et al. 2015]	Discute os principais *desafios na integração entre OpenFlow e LTE*. Dentre eles, destaca a dificuldade em avaliar novas propostas, e um módulo OpenFlow para simulações no *ns-3* é apresentado.
[Li et al. 2012]	*Trabalho motivacional* que aborda as limitações das redes atuais e mostra com o SDN pode substituir os elementos do LTE, tornando a infraestrutura simples com o controle implementado em *software*.
[Arslan et al. 2015]	Investiga como aplicar os conceitos de *SDN nas redes de acesso sem fio*, facilitando a realização de redes auto-configuráveis e a coordenação de interferência entre eNBs.
[Wang et al. 2015]	Apresenta a *arquitetura SoftNet para redes 5G*, que utiliza-se do SDN e NFV para habilitar e desabilitar as funções virtuais da rede, reutilizando os recursos e contribuindo para tecnologias mais limpas.

Tabela 2 (continuação). Diferentes contribuições para redes SDMN flexíveis.

Referência	Contribuição
[Jin et al. 2013]	Propõe a *arquitetura SoftCell*, que substitui o plano de dados por *switches* OpenFlow e funções de rede em *software*. O tráfego é direcionado pelas funções de acordo com o QoS.
[Pentikousis et al. 2013]	Apresenta a *arquitetura MobileFlow*, que desmonta o modelo atual permitindo que as operadoras possam utilizar a mesma infraestrutura para prover diferentes serviços, como acesso 3G e 4G.
[Ali-Ahmad et al. 2013]	Introduz a *arquitetura CROWD* para redes densas, com uma estrutura hierárquica de controladores e realizando agregação de tráfego na borda da rede com foco na escalabilidade.
[Kempf et al. 2012]	Discute como *mover o EPC para um data center*, com detalhe para as mudanças necessárias no protocolo OpenFlow para lidar com roteamento seletivo de tráfego baseado nas aplicações.
[Kyung et al. 2015]	Introduz o conceito de *redes transacionais*, que implementa serviços legados (2G e 3G) sobre a infraestrutura LTE e oferece uma abordagem de transição para as redes futuras.
[Hampel et al. 2013]	Apresenta o conceito de *encaminhamento vertical*, uma estratégia que simplifica o controle de túneis através do des/encapsulamento dos pacotes ao entrar/sair da rede OpenFlow.
[Venmani et al. 2012]	Solução para *controle de congestionamento*, onde a rede de infraestrutura virtualizada pode ser compartilhada entre operadoras, permitindo o controle de congestionamento através da flexibilização das rotas.
[Gurusanthosh et al. 2013]	Mecanismo semi-distribuído de *gerência de mobilidade* para o LTE, onde as âncoras de mobilidade são distribuídas pelos *switches* na rede OpenFlow, em contraste com o modelo atual centrado no S-GW.
[Nagaraj e Katti 2014]	Apresenta o *ProCell*, solução para *balanceamento de carga* que redireciona tráfegos sem QoS restrito do eNB direto para à Internet, desafogando o núcleo EPC.
[Ghazisaeedi e Tafazolli 2013]	Proposta de *balanceamento de carga*, que instala *switches* OpenFlow nos S-GWs para identificar tráfegos HTTP e direcioná-los à Internet, reduzindo a carga sobre o P-GW e sobre os elementos de controle.
[Said et al. 2013]	Solução com foco na *resiliência da rede*, onde a gerência dos túneis GTP passa a ser controlada por *software* centralizado, permitindo rápida recuperação em caso de falha nos equipamentos.
Proposta deste artigo	Mecanismo de *controle de tráfego*, responsável por admissão e roteamento de tráfegos com QoS restritos, que demandam por reserva de banda no núcleo da rede.

4.2. Discussão sobre a flexibilidade oferecida pelo SDN

Dentre as soluções da literatura apresentadas, algumas delas exploram as características do SDN para flexibilizar a gerência dessas redes. É o caso de soluções de controle de congestionamento, balanceamento de carga e resiliência, que fazem uso de *software* para encontrar os pontos críticos da rede e redirecionar o tráfego, explorando o controle centralizado do OpenFlow para flexibilizar essas decisões, que demandariam esforço maior para serem realizadas com as tecnologias tradicionais.

Neste contexto, o mecanismo de controle de tráfego deste trabalho contribui como uma nova estratégia de flexibilização das redes LTE atuais, focado em um novo tópico até então não abordado em outros trabalhos: o controle de admissão em conjunto com o roteamento de tráfegos com reserva de banda. A flexibilidade oferecida pelo SDN é explorada especificamente no mecanismo de roteamento, que permite o uso de duas métricas de roteamento distintas em conjunto. Essa combinação resultou na redução da taxa de bloqueio através do direcionamento seletivo do tráfego por rotas alternativas. O mecanismo proposto tem como características marcantes manter a compatibilidade com o padrão LTE, necessitando apenas da integração em *software* com o MME. Outro ponto notável é a validação da proposta através de simulações no *ns-3*.

5. Conclusões e trabalhos futuros

Neste trabalho foi discutido como o paradigma SDN pode ser aplicado às redes móveis LTE para flexibilizar a arquitetura, aumentando a escalabilidade, reduzindo os custos e permitindo novos serviços. Como principal contribuição foi proposta a integração entre o protocolo OpenFlow e as redes LTE, num modelo em que o controlador comunica-se com o MME para gerenciar o roteamento de túneis sobre a infraestrutura de rede e o controle de admissão de tráfegos que demandam por reserva de banda. Uma topologia de núcleo em anel foi utilizada para validar o mecanismo através de simulações, e os resultados confirmaram os benefícios alcançados com a flexibilidade no roteamento de tráfego oferecido pelo OpenFlow. Também foram analisados outros trabalhos na literatura e tornou-se evidente que a flexibilidade das redes SDMN permite novas soluções, como aquelas voltadas para controle de congestionamento, gerência de mobilidade, balanceamento de carga e redirecionamento de tráfego para desafogar o núcleo da rede. Conclui-se que com o apoio do SDN é possível implantar mais serviços com menos recursos, levando a soluções de longo prazo frente às deficiências atuais.

Como trabalhos futuros, pretende-se explorar o controle de tráfego em redes heterogêneas, como balanceamento de carga entre diferentes tecnologias de acesso e a gerência de mobilidade dos usuários nessas redes. Também pretende-se encontrar métricas que possam ser utilizadas para quantificar a flexibilidade oferecida pelo paradigma SDN em propostas de arquiteturas e soluções para redes integradas, de maneira a corroborar com a discussão qualitativa apresentada neste artigo.

Referências

3GPP 23.203 (2015). Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Policy and charging control architecture. 3GPP TS 23.203, 3GPP.

Ali-Ahmad, H., Cicconetti, C., Oliva, A., Mancuso, V., Sama, M., Seite, P., e Shanmugalingam, S. (2013). An SDN-based network architecture for extremely dense wireless networks. Em *SDN for Future Networks and Services (SDN3FNS)*, pág. 1–7. IEEE.

Arslan, M., Sundaresan, K., e Rangarajan, S. (2015). Software-defined networking in cellular radio access networks: Potential and challenges. *IEEE Communications Magazine*, 53(1):150–156.

Chaves, L., Eichemberger, V., Garcia, I., e Madeira, E. (2015). Integrating OpenFlow to LTE: some issues toward Software-Defined Mobile Networks. Em *Int. Conf. on New Technologies, Mobility and Security (NTMS)*, pág. 1–5. IEEE.

Chaves, L., Garcia, I., e Madeira, E. (2016). OFSwitch13: Enhancing ns-3 with OpenFlow 1.3 support. Em *Workshop on ns-3 (WNS3)*, pág. 1–8. NS Consortium.

Cisco Systems, Inc. (2015). Cisco visual networking index: Global mobile data traffic forecast update, 2014—2019. White Paper.

ETSI NFV (2012). Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. White Paper.

Fitzek, F. e Reisslein, M. (2001). MPEG–4 and H.263 video traces for network performance evaluation. *IEEE Network*, 15(6):40–54.

Ghazisaeedi, E. e Tafazolli, R. (2013). Mobile core traffic balancing by OpenFlow switching system. Em *Globecom Workshops (GC Wkshps)*, pág. 824–829. IEEE.

Gurusanthosh, P., Rostami, A., e Manivasakan, R. (2013). SDMA: A semi-distributed mobility anchoring in LTE networks. Em *Int. Conf. on Mobile and Wireless Networking (MoWNeT)*, pág. 133–139. IEEE.

Hampel, G., Steiner, M., e Bu, T. (2013). Applying software-defined networking to the telecom domain. Em *Conf. on Computer Communications Workshops (INFOCOM Wkshops)*, pág. 133–138. IEEE.

Howard, M. (2011). Using carrier ethernet to backhaul LTE. Infonetics Research White Paper.

Jin, X., Li, L., Vanbever, L., e Rexford, J. (2013). SoftCell: scalable and flexible cellular core network architecture. Em *Conf. on Networking Experiments and Technologies (CoNEXT)*, pág. 163–174. ACM.

Kempf, J., Johansson, B., Pettersson, S., Luning, H., e Nilsson, T. (2012). Moving the mobile evolved packet core to the cloud. Em *Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pág. 784–791. IEEE.

Kyung, Y., Nguyen, T., Hong, K., Park, J., e Park, J. (2015). Software defined service migration through legacy service integration into 4G networks and future evolutions. *IEEE Communications Magazine*, 53(9):108–114.

Li, L., Mao, Z., e Rexford, J. (2012). Toward software-defined cellular networks. Em *European Workshop on Software Defined Networking (EWSDN)*, pág. 7–12. IEEE.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., e Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Nadiv, R. e Naveh, T. (2010). Wireless backhaul topologies: Analyzing backhaul topology strategies. Ceragon White Paper.

Nagaraj, K. e Katti, S. (2014). ProCel: Smart traffic handling for a scalable software EPC. Em *Workshop on Hot Topics in Software Defined Networking (HotSDN)*, pág. 43–48. ACM.

ONF (2012). Software-Defined Networking: The new norms for networks. ONF White Paper.

OpenFlow 1.5.1 (2015). OpenFlow switch specification. Version 1.5.1, Open Networking Foundation.

Pentikousis, K., Wang, Y., e Hu, W. (2013). MobileFlow: Toward software-defined mobile networks. *IEEE Communications Magazine*, 51(7):44–53.

Said, S., Sama, M., Guillouard, K., Suciu, L., Simon, G., Lagrange, X., e Bonnin, J. (2013). New control plane in 3GPP LTE/EPC architecture for on-demand connectivity service. Em *Int. Conf. on Cloud Networking (CloudNet)*, pág. 205–209. IEEE.

Venmani, D., Zeghlache, D., e Gourhant, Y. (2012). Demystifying link congestion in 4G-LTE backhaul using OpenFlow. Em *Int. Conf. on New Technologies, Mobility and Security (NTMS)*, pág. 1–8. IEEE.

Wang, H., Chen, S., Xu, H., Ai, M., e Shi, Y. (2015). SoftNet: A software defined decentralized mobile network architecture toward 5G. *IEEE Network*, 29(2):16–22.

Mecanismos de Alocação Dinâmica de Recursos do RACH para Controle de Congestionamento em rede LTE-A Ocasionado por dispositivos M2M

David B. P. Aragão[1], **Dario Vieira**[2], **Miguel F. de Castro**[1]

[1] Universidade Federal do Ceará (UFC)

[2]Ecole d'Ingénieur Généraliste en Informatique et Technologies du Numérique (EFREI)

{davidbpa,miguel}@great.ufc.br, dario.vieira@efrei.fr

Abstract. *Machine-to-machine (M2M) plays an important role in the paradigm of the Internet of Things. The Long-Term Evolution Advanced (LTE-A) network presents itself, among the wireless networks, as a potential access network for M2M. However, the LTE-A standard also inherits some features from its predecessors that a priori were thought to a Human-to-human (H2H) and human-to-machine (H2M) communication models. Thus, emerge from this integration of the M2M with LTE-A some challenges, with highlight to the overload and congestion problem in the Radio Access Network (RAN) during the Random Access Channel (RACH) procedure. In this article, we propose two mechanisms based on the RACH resource allocation schema for division of channel resources between classes of M2M and H2H devices. Thus, we model the lack of resources as a bankruptcy problem and present two solutions for this problem, one based on the proportional rule and a second one using a framework of cooperative games. The results obtained through extensive simulations in NS-3 simulator show that the mechanisms proposed in this paper present good results in terms of energy efficiency, control of the impact over the H2H devices, the access delay and prioritization between the different classes of devices analyzed.*

Resumo. *O modelo de comunicação Máquina-a-máquina (M2M) é considerado um dos pilares para o paradigma da Internet das Coisas. A rede Long-Term Evolution Advanced (LTE-A) apresenta-se, dentre as redes sem fio, como potencial rede de acesso para o M2M. Contudo, o padrão LTE-A ainda herda algumas características de seus antecessores, que a priori, foram pensados para os modelos de comunicação Humano-a-humano (H2H) e Humano-a-máquina (H2M). Assim, emergem dessa integração do M2M com o LTE-A alguns desafios, com destaque para o problema da sobrecarga e congestionamento na RAN (Radio Access Network) durante o procedimento de RACH (Random Access Channel). Neste artigo são propostos dois mecanismos baseados na estratégia de reserva de recursos do RACH para divisão dos recursos do canal entre as classes de dispositivos M2M e H2H. Para isso, o problema da escassez de recursos é modelado como um problema de falência e posteriormente aplicam-se duas soluções, sendo uma baseada na solução proporcional, e a segunda, utilizando um framework de jogos cooperativos com transferência de utilidade. Os resultados obtidos através de exaustivas simulações no simulador NS-3 mostram que os mecanismos propostos neste trabalho apresentam bons*

resultados em termos de eficiência energética, controle do impacto do sobre os dispositivos H2H, tempo de acesso e priorização entre as diferentes classes de dispositivos consideradas.

1. Introdução

A comunicação Máquina-a-máquina (M2M) viabiliza a troca de informações entre dispositivos sem quase ou nenhuma intervenção humana [Lo et al. 2011]. A comunicação M2M é considerada um dos pilares para viabilização da visão da Internet das Coisas (Internet of Things - IoT), desempenhando papel importante na alavancagem deste paradigma. O LTE é um padrão especificado pela 3GPP (*Third-Generation Partnership Project*), também popularmente conhecido como tecnologia 4G. Devido às características da rede LTE, tais como mobilidade, acessibilidade, vasta área de cobertura e segurança, a rede LTE é apontada como potencial rede de acesso para o desenvolvimento de aplicações e serviços para a Internet das Coisas. Contudo, o padrão LTE, como evolução das redes celulares, herda características de projeto dessas redes (e.g., adoção de um modelo de comunicação entre humanos (H2H) ou entre humanos e máquinas (H2M)). Aproximadamente, 20 bilhões de dispositivos empregando o modelo de comunicação M2M são esperados até 2020. Esses dispositivos serão empregados nas mais diversificadas áreas, tais como: sistemas de medição inteligentes, saúde, segurança e vigilância, dentre outras [Chen et al. 2012]. Devido ao número excessivo de dispositivos M2M esperados para os próximos anos, os problemas da sobrecarga e congestionamento na rede de acesso via rádio (RAN) e núcleo da rede (CN) poderão inviabilizar essa integração. A partir deste contexto, percebe-se a importância de soluções para neutralizar o impacto ocasionado pelo M2M sobre as redes LTE e, consequentemente, possibilitem a integração do M2M [3GPP 2014].

O problema do congestionamento na RAN ocorre essencialmente durante o procedimento de acesso aleatório ao canal (RACH), descrito com mais detalhes na Seção 2. Por comportar-se como o método de acesso ao meio *slotted-aloha*, o aumento do número de dispositivos solicitando acesso à rede, não acompanhado pelo aumento na quantidade de recursos, favorece a sobrecarrega e congestionamento na RAN durante o procedimento de RACH. Ciente deste problema, a 3GPP tem apresentado relatórios (e.g., [3GPP 2014, 3GPP 2012, ZTE 2011, 3GPP 2013]) para ressaltar o problema da sobrecarga e propor possíveis estratégias para neutralizar este problema. Dentre algumas das estratégias apresentadas pela 3GPP, tem-se o ACB (*Access Class Barring*), backoff, separação dos recursos do RACH, alocação dinâmica de recursos do PRACH (*Physical Random Access Channel*), slotted access e o pull access [3GPP 2014]. Embora apresentem algumas desvantagens quando analisados separadamente, tais estratégias ainda são combinadas e aprimoradas por outras abordagens na literatura (e.g., [Lo et al. 2011, Lin et al. 2014, Lee et al. 2011, Aragao et al. 2015]).

Os mecanismos propostos neste trabalho, quando comparado com o trabalho [Aragao et al. 2015], têm em comum o objetivo de reduzir o congestionamento na RAN e suas consequências, mas diferem em relação as técnicas utilizadas para atingir tal objetivo. Em [Aragao et al. 2015], a estratégia de controle atua na camada de acesso ao meio dos dispositivos ao passo que as propostas apresentadas neste trabalho agem na estação base (eNodeB - *evolved NodeB*). Com isso, as propostas apresentadas neste trabalho têm um caráter mais incisivo na determinação de como os recursos devem ser utilizados. Vale

ressaltar que os mecanismos proposto neste trabalho não assumem um caráter excludente em relação ao trabalho [Aragao et al. 2015], mas de complementação. A Seção 3, apresenta os trabalhos relacionados.

O problema da sobrecarga e congestionamento motivou o desenvolvimento deste trabalho. Neste trabalho são propostos dois mecanismo para controlar o impacto gerado pelos dispositivos M2M sobre os dispositivos H2H durante o procedimento de RACH. Os mecanismos propostos neste trabalho se baseiam na estratégia de reserva de recursos do RACH da 3GPP para os dispositivos H2H e M2M [3GPP 2014]. Nessa estratégia, os recursos reservados para o procedimento de RACH são alocados para os diferentes tipos de dispositivos, restringindo as colisões aos dispositivos de uma mesma classe. Contudo, a estratégia apresentada pela 3GPP assume um comportamento estático, ou seja, desconsiderando o nível de congestionamento na RAN. Neste trabalho, os mecanismos propostos flexibilizam a estratégia apresentada pela 3GPP permitindo que a reserva dos recursos seja realizada dinamicamente. Além disso, os mecanismos propostos neste trabalho também consideram a priorização entre os dispositivos M2M com a criação de diferentes classes de prioridade. As propostas deste trabalho são detalhadas na Seção 4. A motivação pela criação das diferentes classes de dispositivos M2M está associada a possibilidades de estender a prioridade entre as aplicações e serviços alicerçados sobre as redes LTE durante o procedimento de RACH. Os resultados obtidos através de exaustivas simulações realizadas no NS-3, apresentados e discutidos na Seção 5, mostram que os mecanismos propostos neste trabalho alcançam os objetivos supracitados.

O restante deste trabalho está organizado na seguinte forma. A Seção 5 apresenta o cenário de simulação. Finalmente, a Seção 6 apresenta a conclusão e trabalhos futuros.

2. Fundamentação Teórica

2.1. Procedimento de Acesso Aleatório ao Canal (RACH)

O procedimento de RACH do LTE é iniciado, dentre outras razões, para o estabelecimento de conexão entre o dispositivo e a rede e pode ser categorizado em livre de contenção ou baseado em contenção. No método livre de contenção a comunicação é iniciada pelos elementos do núcleo da rede e está associado às atividades que apresentam fortes restrições de tempo (e.g., *handover*). No método baseado em contenção a comunicação é iniciada pelos dispositivos e estes competem entre si pelos recursos do canal para enviarem suas solicitações. O procedimento de RACH baseado em contenção é composto por quatro mensagens (msg1, msg2, msg3, msg4) que são trocadas entre o dispositivo e a estação base, como ilustrado na Fig. 1. As seguintes atividades são realizadas durante cada etapa da troca de mensagem:

1. Envio do preâmbulo de PRACH - *PRACH Preamble* (msg1): O dispositivo escolhe aleatoriamente um código de preâmbulo e aguarda o próximo *slot* de acesso (*RA-slot*) para enviar sua solicitação de acesso.
2. Resposta da Solicitação de Acesso - *Random Access Response* (msg2): As seguintes atividades são realizadas neste estágio: (i) detecção de recebimento da mensagem pela eNodeB, (ii) assinalamento de um identificador temporário para o dispositivo e (iii) atribuição de recursos para o dispositivo no canal de *uplink*.
3. Solicitação de Contenção - *Contention Request* (msg3): O dispositivo envia o seu identificador para a estação base no canal compartilhado de *uplink* alocado no estágio anterior do procedimento de RACH.

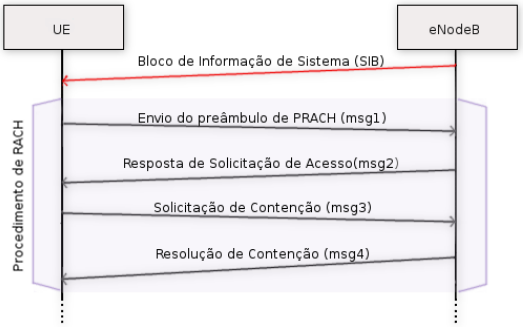


Figura 1. Etapas do Procedimento de RACH Baseado em Contenção

4. Resolução de Contenção - *Contention Resolution* (msg4): Após o recebimento da mensagem, o dispositivo verifica se nela consta o seu identificador. Se sim, o dispositivo envia uma mensagem de confirmação. Caso contrário, o dispositivo entra no estado de espera cujo tempo é definido pelo indicador de *backoff* (*Backoff Indicator* - (BI)) recebido da eNodeB. Durante o período de espera o dispositivo fica impossibilitado de enviar novas solicitações de acesso.

2.2. Teoria dos Jogos

A teoria dos jogos é um conjunto de ferramentas utilizadas para modelagem e compreensão da interação entre indivíduos, também chamados de jogadores, diante de cenários de tomada de decisão. São assertivas comuns à teoria dos jogos o comportamento racional dos participantes do jogo e a tomada de decisão realizada pelos seus jogadores a partir do conhecimento e comportamento esperado dos outros participantes [Osborne and Rubinstein 1994]. Os principais elementos presentes em um jogo são: jogadores/indivíduos, estratégias/ações e retorno/pagamento (*payoff*).

O jogador é a entidade básica comum a todos os modelos de jogos. Dependendo do tipo de jogo, a interação dos jogadores durante o jogo pode se analisada a partir de duas perspectivas. Na primeira, a interação entre os jogadores pode ser analisada individualmente, ou seja, cada jogador age com o intuito de maximizar o seu retorno e tendem a não cooperar. Na segunda, comum aos jogos cooperativos, a análise da interação deixa de ser individualizada para ser sobre o grupo de jogadores, ou seja, os jogadores tendem a cooperar formando coalizão para maximizar o retorno/pagamento. Nos jogos cooperativos não existe a noção de conjunto de estratégias individuais. Os jogos cooperativos podem ser classificados em jogos com transferência de utilidade e jogos sem transferência de utilidade. Nos jogos com transferência de utilidade, as ações tomadas pelas coalizões determinam o *payoff* para cada coalizão. Nos jogos sem transferência de utilidade, as ações tomadas pela coalizão determinam o *payoff* de cada indivíduo.

2.3. Problema de Falência

O problema da falência é utilizado para descrever cenários de falência, ou seja, cenários onde a quantidade de recursos disponíveis para serem distribuídos entre os indivíduos é menor do que o somatório das reivindicações dos indivíduos [Herrero and Villar 2001]. Diante deste cenário, surge o problema de como esses recursos devem ser distribuídos entre os indivíduos de modo justo. Dentre as soluções amplamente conhecidas, tem-se a

regra proporcional e a estratégia utilizando jogos cooperativos com transferência de utilidade (TU) [Herrero and Villar 2001]. Na regra proporcional, o senso de justiça é definido a partir das diferentes características desejadas durante a divisão dos recursos e essas características são, por sua vez, descritas através de axiomas. O senso de justiça presente na solução utilizando jogos cooperativos é herdado do Valor de Shapley [Shapley 1952], explicado ainda nesta seção, e dos axiomas que embasam a sua definição de justiça.

O problema de falência podem ser modelados da seguinte forma: seja $N = \{1, 2, 3, \ldots, n\}$ um conjunto finito com todos os indivíduos, grupos/classes. Um problema de falência para N pode representado pela tupla (E, c), onde $E \in \mathbb{R}_+$ identifica a quantidade de recursos disponíveis para serem distribuídos entre os N indivíduos e $c \in \mathbb{R}^n_+$ o vetor com as reivindicações dos indivíduos. Deste modo, c_i identifica a quantidade de recursos demandados pelo i-ésimo indivíduo.

Na regra proporcional, o problema de falência pode ser representado do seguinte modo: $P(E, c)$, onde E representa a quantidade de recursos disponíveis e c, como apresentado anteriormente, também representa o vetor de reivindicações dos indivíduos. Assim, para todo problema de falência representado por (E, c), existe uma solução $P(E, c) = \lambda c$, $\lambda = \frac{E}{C}$, onde $C = \sum_{i \in N} c_i$, onde N representa os indivíduos. São características sustentadas pela regra proporcional: (i) tratamento igual de iguais; (ii) invariância de escala ou homogenia de grau um; (iii) composição; (iv) aditividade; (v) monotonicidade.

Uma das soluções utilizando jogos para o problema de falência está apresentada em [O'Neill 1982], que modela o problema usando uma estratégia de jogos cooperativos com transferência de utilidade. Um jogo cooperativo com TU aplicado ao problema da falência pode ser representado a partir da tupla $G(N, v)$, onde N representa os indivíduos e v a função característica, $v : 2^n \rightarrow \mathbb{R}$ e $v(\emptyset) = 0$. A cooperação entre os indivíduos em um jogo cooperativo inexiste na etapa que precede a aquisição dos recursos pela coalizão. Um desafio comum aos jogos cooperativos é como os recursos alocados devem ser distribuídos entre os indivíduos que formam a coalizão. Uma coalizão formada por todos os indivíduos recebe o nome de grande coalizão. A fórmula para cálculo do valor de Shapley, apresentada em 1953 por Lloyd Shapley, é considerada uma abordagem justa para divisão desses recursos entre os indivíduos da coalizão. As propriedades que definem a justiça do valor de Shapley são: (i) eficiência; (ii) simetria; (iii) agregação; (iv) jogador nulo. Como apresentado em [Shapley 1952], o cálculo da contribuição de um dado jogador i a um jogo $G(N, v)$ é dado por: $\phi_i(v) = \sum_{S \subseteq \mathbb{N} \setminus i} \frac{|S|!(n-|S|-1)!}{n!}(v(S \cup \{i\}) - v(S))$, onde $S \subseteq N$.

3. Trabalhos Relacionados

Em [Pang et al. 2014], os autores propõem uma estratégia para divisão dos códigos de preâmbulos do RACH em três grupos: H2H, M2M e um grupo híbrido. Os preâmbulos de cada grupo só podem ser utilizados pelos dispositivos que fazem parte do grupo. Contudo, os preâmbulos alocados para o grupo híbrido podem ser utilizados pelos dispositivos das classes H2H e M2M. O foco principal do trabalho é definir como os recursos do grupo híbrido devem ser compartilhados entre os dispositivos H2H e M2M. Para isso, os autores modelam este cenário de decisão como um jogo não cooperativo. Contudo, a alocação

fixa de recursos para os dispositivos H2H e M2M impossibilita que os recursos sejam realocados de acordo com a necessidade de cada grupo. O cenário de simulação não é suficientemente congestionado o que dificulta a análise do mecanismo diante de um cenário congestionado. Além disso, os autores não estabelecem priorização entre os dispositivos M2M.

Em [Lee et al. 2011], os autores apresentam dois métodos para divisão dos recursos entre os dispositivos das classes H2H e M2M. No primeiro, os preâmbulos são divididos em grupos disjuntos para os dispositivos H2H e M2M. O segundo método difere do primeiro em relação à utilização dos recursos alocados para o M2M, pois nesse método os preâmbulos, antes alocados somente para os dispositivos M2M, também podem ser utilizados pelos dispositivos H2H. Os dois métodos utilizados para dividir os recursos fixam a quantidade de recursos que devem ser alocados. Os autores não estabelecem nenhuma priorização entre os dispositivos M2M.

Em [Jian et al. 2013], os autores se baseiam em uma abordagem centrada na técnica de backoff, onde diferentes intervalos de espera são atribuídos aos diferentes tipos de dispositivos. Na ocorrência de colisões, cada dispositivo aguarda por um determinado intervalo de tempo, de acordo com a sua classe de backoff, antes de retransmitir uma nova solicitação. A principal desvantagem apresentada por este mecanismo está na estratégia utilizada para definir o intervalo das classes de backoff, pois não estabelecem nenhuma relação entre o tempo atribuído as classes de backoff e o número de dispositivos em cada classe. Ou seja, não importa quantos dispositivos há em cada classe, o intervalo atribuído às classes permanece inalterado.

Em [Choi et al. 2011], os autores propõem uma estratégia para alocação dinâmica de recursos do PRACH para reduzir a latência no acesso dos dispositivos. A técnica de alocação dinâmica de recursos do PRACH baseia-se na semialocação de recursos para o envio de códigos de preâmbulos no PRACH durante o procedimento de RACH. A relação entre os *slots* para envio dos preâmbulos de acesso (*RA-slots*) e a latência no acesso é apresentada como um problema de otimização. Para auxiliar na solução deste problema, os autores propõem um modelo para identificar a taxa de chegada das solicitações de acesso à rede. Contudo, a semialocação de códigos de preâmbulos para o PRACH acarreta redução nos recursos que antes estavam alocados para a transferência de dados. Os autores não analisam como a estratégia pode impactar na transferência de dados no canal de *uplink*. Além disso, os autores não estabelecem nenhuma priorização entre os dispositivos M2M.

Em [Lo et al. 2011], os autores propõem um mecanismo para controlar a sobrecarga durante o procedimento de RACH que utiliza a técnica de alocação dinâmica de recursos para o envio dos códigos de preâmbulos do PRACH. A quantidade de recursos disponibilizados varia de acordo com o nível de congestionamento na RAN. O nível de congestionamento na RAN é inferido a partir do número de tentativas de acesso já enviadas pelos dispositivos. O número de tentativas é informado para a estação base por cada dispositivo quando este se conecta a rede. Além da técnica de alocação dinâmica de recursos, os autores utilizam as técnicas *Slotted-Access* e ACB para mitigar o impacto sobre os dispositivos H2H gerado pelo excesso de dispositivos M2M e prover priorização entre os dispositivos M2M. Infelizmente o mecanismo proposto não é avaliado pelos autores, seja por simulação ou analiticamente.

Os mecanismos propostos neste trabalho utilizam a estratégia de divisão dos recursos do RACH entre dispositivos das classes: (i) H2H, (ii) M2M de alta e (iii) M2M de baixa prioridade. A diferenciação entre dispositivos M2M de baixa e alta prioridade é influenciada pela possível variação nos requisitos existentes entre as aplicações e serviços IoT. Além disso, este trabalho também analisa a eficiência energética dos mecanismos propostos. A modelagem do problema da escassez de recursos do RACH como um problema de falência e a estratégia utilizada para composição do vetor de reivindicações são alguns dos aspectos que diferenciam este trabalho das outras abordagens apresentadas.

4. Abordagens Propostas

Neste artigo, o problema da divisão dos recursos do RACH é modelado como um problema de falência. As diferentes classes de dispositivos são representadas por N, a quantidade de recursos disponíveis é dado por E. A reivindicação de cada classe de dispositivo é armazenada no vetor de reivindicações c_i, onde $i = \{1, 2, \ldots, n\}$ e $n = |N|$. Os mecanismos propostos neste trabalho se baseiam nas soluções que abordam o problema a partir de uma estratégia de jogos cooperativos com transferência de utilidade e na regra proporcional.

Durante o procedimento de RACH cada dispositivo escolhe aleatoriamente um código de preâmbulo e aguarda o próximo *slot* de acesso (*RA-Slot*) disponível para enviar sua solicitação para a eNodeB. Contudo, com o aumento do número de solicitações mais colisões tendem a ocorrer, aumentando nível de congestionamento na RAN. Considerando que há M códigos de preâmbulos disponíveis para o procedimento de RACH baseado em contenção e K dispositivos competindo por esses recursos. O número esperado de códigos de preâmbulos recebidos com sucesso pela estação base (Y), ou seja, escolhidos por somente um dispositivo é dado:

$$E[Y] = K(1 - \frac{1}{M})^{K-1} \tag{1}$$

Assim, o número de preâmbulos necessários para suprir a demanda de K dispositivos que estão solicitando acesso à rede é dado por: $F(M) = K \times \frac{1}{\sigma}$, onde σ é o desempenho máximo esperado do canal (cf., Eq. 1).

4.1. Vetor de Reivindicações

Esta seção identifica como a eNodeB calcula o vetor de reivindicação. Seja N as diferentes classes de dispositivos. Cada classe solicita $M_i = Col_i + L_i$ códigos de preâmbulos, onde $i \in N$, Col_i identifica a quantidade de solicitações que colidiram e L_i os preâmbulos recebidos com sucesso. Baseado na Eq. 1, o número esperado de solicitações recebidas com sucesso pela eNodeB é dada por: $E[Y] = \sum_{i \in N} K_i(1 - \frac{1}{M_i})^{K_i - 1}$, onde K_i identifica a quantidade de dispositivos de uma dada classe i. O vetor que armazena as reivindicações das i classes de dispositivos é dado por: $c_i = \frac{1}{j} \sum_{l \in 0}^{j} F(L_{i,k-l} + Col_{i,k-l})$, onde k identifica o procedimento de RACH corrente. Essa equação calcula a média móvel simples (SMA) dos preâmbulos recebidos e as colisões que ocorreram durante as últimas j tentativas de acesso. Neste trabalho, a priorização entre as classes de dispositivos é realizada a partir da atribuição de pesos ao vetor de reivindicações das diferentes classes de dispositivos. Além disto, assume-se que a prioridade dos dispositivos da classe H2H é maior do

que a dos dispositivos da classe M2M. Os dispositivos M2M também são divididos em dispositivos de alta e baixa prioridade.

Os pesos (α, β e γ) atribuídos às diferentes classes de dispositivos foram definidos a partir de simulações. A priorização entre os dispositivos durante a simulação foi mensurada a partir do número total de acessos. Os pesos são atribuídos de forma a garantir que o total de acessos realizados pelos dispositivos M2M prioritários (alta prioridade) seja de aproximadamente 50% em relação ao H2H, e 100% em relação aos dispositivos M2M não prioritários (baixa prioridade). O calculo da quantidade total de recursos demandado pelas n classes de dispositivos é dada por: $C = \alpha(\frac{1}{j}\sum_{l\in 0}^{j} F(M_{H2H,k-l})) + \beta(\frac{1}{j}\sum_{l\in 0}^{j} F(M_{M2M_p,k-l})) + \gamma(\frac{1}{j}\sum_{l\in 0}^{j} F(M_{M2M_{wp},k-l}))$. Os pesos utilizados para os mecanismos propostos neste trabalho são 3,2,1 e 3,2,0.265, sendo estes para a solução utilizando jogos cooperativos e aqueles para a solução baseada na regra proporcional. Além dos pesos que são atribuídos ao vetor de reivindicações, também há a priorização no atendimento das solicitações pela própria estação base. Nas propostas apresentadas neste trabalho, os códigos de preâmbulos são divididos dinamicamente entre as diferentes classes de dispositivos.

4.2. Alocação de Códigos de Preâmbulos

Neste trabalho, os dispositivos M2M foram divididos em classe de alta prioridade e classe de baixa prioridade. Assim, os indivíduos competindo pelos recursos pertencem as classes H2H, M2M de alta prioridade (M2M$_p$) e M2M de baixa prioridade (M2M$_{wp}$), tal que $N = \{H2H, M2M_p, M2M_{wp}\}$. Os recursos disponíveis para serem distribuídos entre as diferentes classes são os códigos de preâmbulos definidos para o procedimento de RACH. Neste trabalho, a coalizão é igual ao número de indivíduos, ou seja, tem-se uma grande coalizão formada pelas três classes de dispositivos. A partir deste cenário de falência, as seguintes abordagens são propostas:

- **Proporcional**: define a quantidade de códigos de preâmbulos que serão alocados para cada uma das classes de dispositivos usando a regra $P(E, c)$, onde $E = T$, sendo T a quantidade de códigos de preâmbulos disponíveis para o procedimento de RACH baseado em contenção e c o vetor com as reivindicações das classes, apresentado na Seção 4.1. Neste trabalho, este mecanismo será referenciado por 'Proporcional'.
- **Shapley**: define a quantidade de códigos de preâmbulos que serão alocados para cada uma das classes a partir do jogo $G(N, v)$, onde $v(S) = max\{T - \sum_{i\in N\setminus S} c_i, 0\}$ e $S \subseteq N$ o conjunto das possíveis coalizões [O'Neill 1982]. A função característica ($v(S)$) valora a relevância de uma coalizão a partir do resultado da subtração dos recursos disponíveis menos os recursos solicitados pelas possíveis coalizões. O cálculo da quantidade de preâmbulos que serão alocados para cada uma das classes é realizado a partir do Valor de Shapley $\phi_i(v)$, apresentado na Seção 2.3. Neste trabalho, este mecanismo será referenciado por 'Shapley'.

As propostas apresentadas neste trabalho são comparadas com uma abordagem fixa, ou seja, onde os códigos de preâmbulos são fixados para cada classe de dispositivo.

5. Ambiente de Simulação e Resultados Numéricos

Os mecanismos propostos neste trabalho foram validados no simulador de rede NS-3 [NS-3 2014]. Os dispositivos são categorizados em H2H, M2M de alta prioridade

($M2M_p$) e M2M de baixa prioridade ($M2M_{wp}$), e estão solicitando acesso a uma única eNodeB. A eNodeB está configurada para operar na largura de banda de 5 MHz. O índice de configuração do PRACH define quantos *slots* de acesso (*RA-slots*) são disponibilizados para os dispositivos enviarem suas solicitações. A taxa de chegada dos dispositivos segue uma distribuição de Poisson ($\lambda_{H2H} = 1/300$, $\lambda_{M2M_p} = 1/200$ e $\lambda_{M2M_{wp}} = 1/500$). O número de dispositivos H2H está fixado em 200, enquanto o número de dispositivos $M2M_p$ e $M2M_{wp}$ variam entre $\{50, 200, 450, \ldots, 950\}$ e $\{100, 400, 700, \ldots, 1900\}$, respectivamente. Cada cenário de simulação foi executado 30 vezes, e o intervalo de confiança considerado foi de 95%. O cenário de simulação supracitado está sumarizado na Tabela 1 e foi baseado nas configurações apresentadas pela 3GPP em [3GPP 2014].

Parâmetros Gerais da Rede	
Largura de Banda	5 MHz (25 RBs)
Número de Estações Base (eNB)	1
Índice de Configuração do PRACH	6 (n = 2)
Número de Máximo de Retransmissões de Preâmbulo (L)	10
Tempo da Janela de Resposta (RAR)	5 ms
Códigos de Preâmbulo Disponíveis	54
Intervalo de Chegada dos Dipositivos	[0,...,1000] ms
Tempo de Simulação	1000 ms
Dispositivos H2H	200
Dispositivos M2M de alta prioridade ($M2M_p$)	{50,200,350,...,950}
Dispositivos M2M de baixa prioridade ($M2M_{wp}$)	{100,400,700,...,1900}
Taxa de Chegada (H2H) - Poisson	λ_{H2H} = 1/300
Taxa de Chegada ($M2M_p$) - Poisson	λ_{M2M_p} = 1/200
Taxa de Chegada ($M2M_{wp}$) - Poisson	$\lambda_{M2M_{wp}}$ = 1/500
Parâmetros dos mecanismos Shapley (jogo cooperativo) e Proporcional (regra proporcional)	
Recursos Disponíveis: 54 códigos; σ = 0.37; $j = 4$; SIB2 = 10 ms(periodicidade), δ = 0.37; Pesos $\alpha = 3$, $\beta = 2$, $\gamma = 1$ - Proporcional / $\alpha = 3, \beta = 2, \gamma = 0.265$ - Shapley	

Tabela 1. Parâmetros utilizados nas simulações

5.1. Indicadores Chave de Performance

Os indicadores utilizados para analisar as propostas foram: o tempo de acesso, a probabilidade de acesso, o número de acessos realizados com sucesso e número total de preâmbulos transmitidos. O tempo de acesso é o tempo transcorrido entre o envio da primeira solicitação de acesso até o instante em que o dispositivo consegue acessar a rede. O número de acessos é dado pela média de acessos realizados com sucesso dentro do intervalo de simulação considerado. Os três mecanismos simulados neste trabalho serão referenciados por: 'Fixo', 'Proporcional' e 'Shapley', onde 'Fixo' identifica a estratégia de alocação fixa, 'Proporcional' o mecanismo baseada na regra proporcional e 'Shapley' o mecanismo baseado no valor de Shapley.

5.2. Impacto Sobre os Dispositivos H2H

O impacto sobre o número total de dispositivos H2H que conseguem acessar a rede com sucesso é muito baixo. Independente do número de dispositivos M2M, quase todos os dispositivos H2H conseguem acessar a rede, como ilustrado na Fig. 2. Por apresentarem prioridade superior à dos demais dispositivos, os dispositivos da classe H2H têm suas reivindicações mais facilmente atendidas. Além disso, vale ressaltar a priorização que também é dada aos dispositivos desta classe pela eNodeB (cf., Seção 4.1). A alta prioridade definida para os dispositivos H2H também reflete na probabilidade de acesso, que se mantém dentro do intervalo de 1.0 a 0.995, como apresentado na Fig. 2. Em suma, as três abordagens controlam o impacto dos dispositivos M2M sobre os dispositivos H2H.

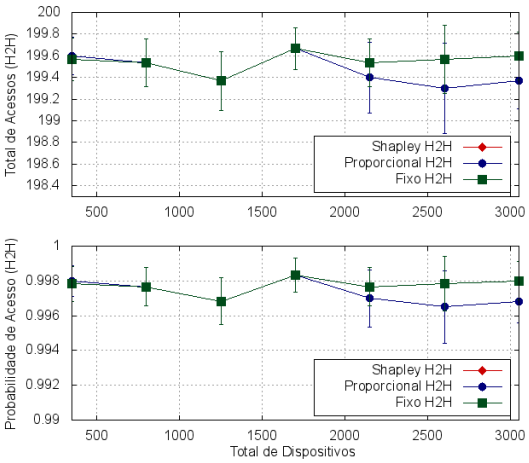
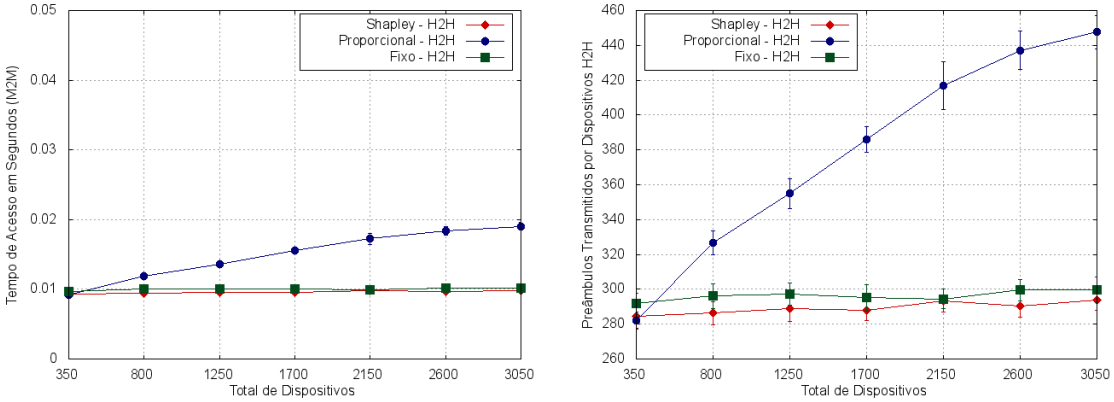


Figura 2. Total de acessos e probabilidade de acesso



(a) Tempo de acesso (b) Número de preâmbulos transmitidos

Figura 3. Tempo de acesso e total de preâmbulos transmitidos

O tempo de acesso aumenta à medida que mais dispositivos solicitam acesso a rede, como ilustrado na Fig. 3(a). Entre as três estratégias simuladas, as abordagens Shapley e Fixo apresentam resultados melhores do que a abordagem Proporcional. Contudo, mesmo apresentando desempenho inferior aos mecanismos Shapley e Fixo, o mecanismo Proporcional ainda sustenta o tempo de acesso abaixo 0.02 s.

O número total de preâmbulos transmitidos por cada classe de dispositivo está ilustrado na Fig. 3(b). Para o cenário com 3050 dispositivos, o mecanismo proporcional apresenta o pior resultado, onde aproximadamente 450 preâmbulos são transmitidos. Resultados melhores são apresentados pelos mecanismos Fixo e Shapley, onde são transmitidos aproximadamente 290 e 300 preâmbulos, respectivamente. Com exceção dos cenários onde há menos de 500 dispositivos solicitando acesso, o desempenho apresentado pelos mecanismos Fixo e Shapley se mantém melhor do que o mecanismo proporcional.

5.3. Impacto Sobre os Dispositivos M2M

A Fig. 4 identifica como os mecanismos reagem ao aumento do número dos dispositivos M2M de alta e baixa prioridade. A partir dos resultados apresentados na Fig. 4 é possível verificar que o mecanismo Proporcional apresenta desempenho melhor do que os outros

mecanismos simulados. Além de possibilitar que mais dispositivos M2M acessem a rede, percebe-se que o mecanismo Proporcional reage mais rapidamente às solicitações dos dispositivos M2M de alta prioridade, como identificado pelo ponto 'A' na Fig. 4. A inversão que ocorre nos pontos 'A' e 'B' está associada à prioridade estabelecida para os dispositivos M2M prioritários em relação aos de baixa prioridade. Assim, quando mais dispositivos M2M prioritários estão solicitando acesso, a estação base passa a realocar os recursos do M2M de baixa prioridade para o M2M prioritário.

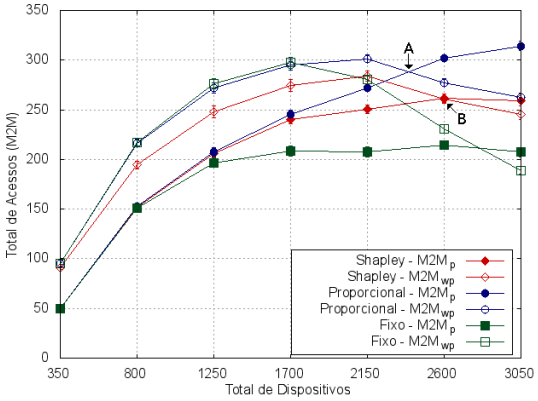
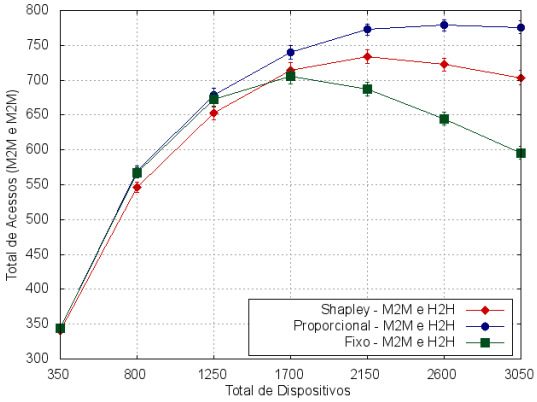
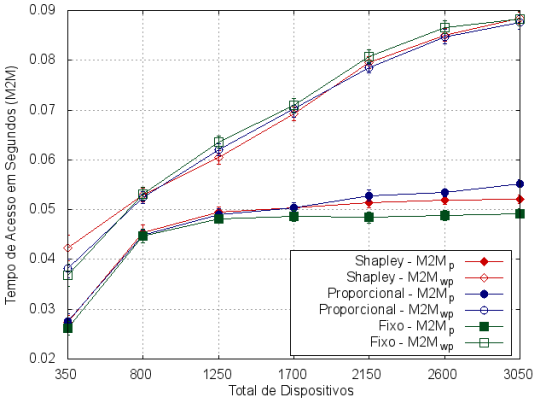


Figura 4. Acessos realizados com sucesso pelos dispositivos $M2M_p$ e $M2M_{wp}$

A Fig. 5(a) apresenta a soma dos acessos realizados pelas classes H2H, M2M$_p$ e M2M$_{wp}$ dos três mecanismo simulados neste trabalho. Como apresentado na Fig. 2, a quantidade de dispositivos H2H que conseguem efetuar acesso com sucesso permanece praticamente constante entre os três mecanismos. Assim, o desempenho dos mecanismos Proporcional e Shapley em relação a implementação fixa deve-se essencialmente aos dispositivos da classe M2M. Como apresentado na Fig. 5(a) os mecanismos Fixo, Shapley e Proporcional atingem performance máxima próxima de 700, 740 e 780 acessos, respectivamente. Depois de atingidos esses picos, o número de acessos realizados com sucesso começa a reduzir, com queda mais acentuada apresentada pelo mecanismo fixo.



(a) Total de acesso efetuados com sucesso pelos pispositivos H2H, $M2M_p$ e $M2M_{wp}$



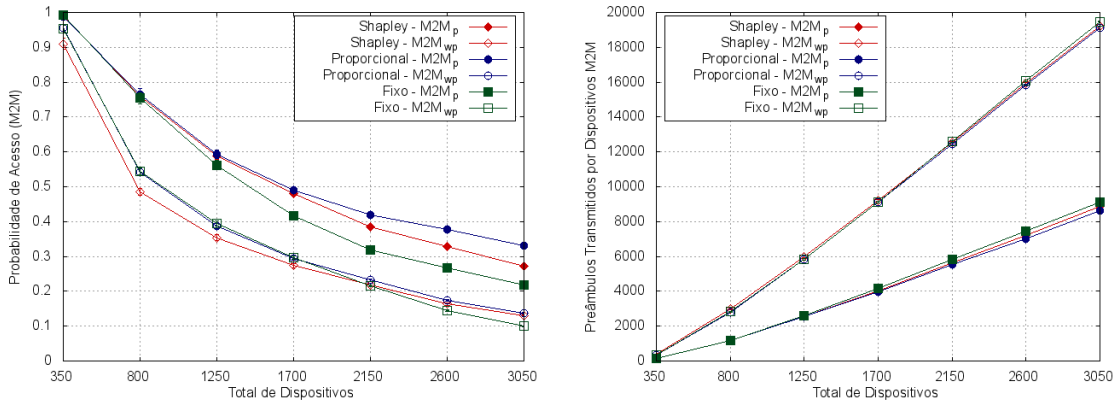
(b) Tempo de acesso dos dispositivos $M2M_p$ e $M2M_{wp}$

Figura 5. Total de acessos e o tempo de acesso

O tempo médio de acesso apresentado pelos mecanismos para os dispositivos das

classes M2M$_{wp}$ e M2M$_{p}$ é de respectivamente 0.09 e 0.055, como ilustrado na Fig. 5(b). Esses resultados embasam as vantagens apresentadas pelos mecanismos propostos neste trabalho em relação ao mecanismo Fixo. Os mecanismos propostos neste trabalho possibilitam que mais dispositivos acessem a rede sem que, para isso, tenham que aumentar o tempo de acesso.

Os resultados ilustrados na Fig. 6(a) mostram como cada estratégia lida com a priorização entre os dispositivos M2M. Quando analisados separadamente, é possível verificar que os mecanismos simulados proveem priorização entre os dispositivos M2M$_{p}$ e M2M$_{wp}$. Quando comparados entre si, a priorização estabelecida pelos mecanismos Porporcional e Shapley para a classe de dispositivo M2M prioritário é mais evidente do que a provida pelo mecanismo Fixo. Em relação aos mecanismos propostos, o Proporcional apresenta desempenho melhor do que o Shapley em relação a priorização, como evidenciado na Fig. 6(a). Em relação os dispositivos da classe M2M$_{wp}$, percebe-se que a diferença entre os mecanismos é pouco significativa, como ilustrado na Fig. 6(a).



(a) Probabilidade de acesso

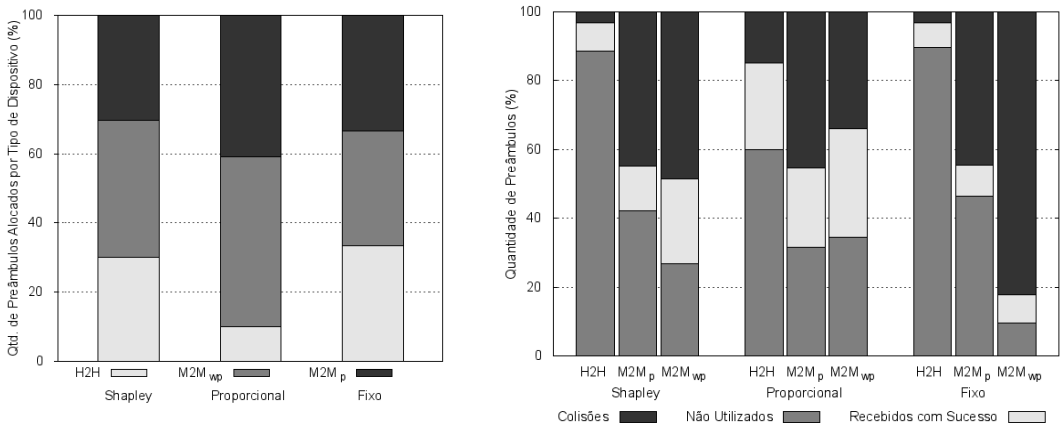
(b) Número Total de preâmbulos

Figura 6. Probabilidade de acesso e total de preâmbulos transmitidos

A quantidade de preâmbulos transmitidos pelos três mecanismos é aproximadamente igual nas três abordagens simuladas, como apresentado pela Fig. 6(b). Contudo, o número de preâmbulos recebidos com sucesso, ou seja, sem colisão é maior nos mecanismos Proporcional e Shapley, como apresentado na Fig. 7(b). Por ser possível inferir causalidade entre número de preâmbulos e eficiência energética, os mecanismos Proporcional e Shapley podem ser considerados energeticamente mais eficientes.

A Fig. 7(b) complementa a informação apresentada na Fig. 7(a), mostrando como os códigos de preâmbulos estão distribuídos para o cenário de simulação com 3050 dispositivos. O mecanismo Fixo, como esperado, distribui os recursos igualmente entre as três classes de dispositivos, como ilustrado na Fig. 7(a). Assim, é possível observar que muitos dos recursos alocados para o H2H não estão sendo utilizados, como ilustrado na Fig. 7(b). Contudo, ao mesmo tempo em que há códigos sobrando para os dispositivos da classe H2H, tem-se os dispositivos M2M$_{wp}$ sofrendo as consequências da escassez de recursos, como pode ser evidenciado pelo número excessivo de colisões.

Comparado com a abordagem fixa, o Shapley distribui mais recursos para os dispositivos M2M de baixa prioridade. A Fig. 7(b) ilustra como os recursos estão sendo



(a) Distribuição dos recursos entre as classes

(b) Utilização dos preâmbulos pelas classes

Figura 7. Utilização dos preâmbulos

distribuídos pelo Shapley. É possível observar que o Shapley apresenta um desempenho melhor do que o Fixo em relação aos dispositivos da classe M2M. Além disso, percebe-se controle sobre as colisões para os dispositivos H2H. Contudo, o número de preâmbulos não utilizados pelos dispositivos H2H permanece elevado no mecanismo Shapley.

Na abordagem Proporcional mais recursos são alocados para os dispositivos da classe M2M. Os resultados, como ilustrado na Fig. 7(b), mostram um gráfico mais balanceado, mas que apresenta muitas colisões. As consequências em reduzir o número total de preâmbulos alocados para os dispositivos H2H refletem no resultado previamente mencionado, ou seja, aumento no tempo de acesso e número de preâmbulos transmitidos pelos dispositivos da classe H2H.

6. Conclusões

Neste artigo são apresentados dois mecanismos para controlar o congestionamento ocasionado na RAN da rede LTE devido ao excesso de dispositivos M2M solicitando acesso à rede. Os mecanismos propostos baseiam-se no esquema apresentado pela 3GPP para separação dos códigos de preâmbulos entre as três classes de dispositivos, sendo elas: H2H, M2M de alta prioridade e M2M de baixa prioridade. Neste trabalho, o problema da alocação de códigos de preâmbulos é modelado como um problema de falência, onde os códigos de preâmbulos disponíveis não são suficientes para satisfazer a quantidade de códigos demandado por cada classe. Os resultados obtidos através de exaustivas simulações mostram que todos os mecanismos simulados controlam o impacto dos dispositivos M2M sobre os dispositivos H2H e a priorização entre os dispositivos M2M. Contudo, os mecanismos propostos apresentam desempenho melhor do que o mecanismo Fixo em termos de eficiência energética, número total de acessos e tempo de acesso. Figuram dentre as atividades a serem realizadas em trabalhos futuros: (i) aprimoramento da estratégia utilizada para definir o vetor de reivindicações; (ii) emprego da técnica de alocação de códigos de preâmbulos com outras estratégias (i.e., [Aragao et al. 2015]).

Referências

3GPP (2012). System improvements for machine-type communications. Technical report, 3GPP.

3GPP (2013). Service requirements for machine-type communications (mtc). Technical report, 3GPP.

3GPP (2014). Study on ran improvements for machine-type communication. Technical report, 3GPP.

Aragao, D. B., Vieira, D., and de Castro, M. F. (2015). Uma proposta de controle de congestionamento ocasionado pela comunicaç ao máquina-a-máquina em lte. In *Proceedings of the 32th Brazilian Symposium on Computer Networks (SBRC 2015)*.

Chen, M., Wan, J., and Li, F. (2012). Machine-to-machine communications. *KSII Transactions on Internet and Information Systems (TIIS)*, 6(2):480–497.

Choi, S., Lee, W., Kim, D., Park, K.-J., Choi, S., and Han, K.-Y. (2011). Automatic configuration of random access channel parameters in lte systems. In *Wireless Days (WD), 2011 IFIP*, pages 1–6.

Herrero, C. and Villar, A. (2001). The three musketeers: four classical solutions to bankruptcy problems. *Mathematical Social Sciences*, 42(3):307–328.

Jian, X., Jia, Y., Zeng, X., and Yang, J. (2013). A novel class-dependent back-off scheme for machine type communication in lte systems. In *Wireless and Optical Communication Conference (WOCC), 2013 22nd*, pages 135–140. IEEE.

Lee, K.-D., Kim, S., and Yi, B. (2011). Throughput comparison of random access methods for m2m service over lte networks. In *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, pages 373–377. IEEE.

Lin, T.-M., Lee, C.-H., Cheng, J.-P., and Chen, W.-T. (2014). Prada: Prioritized random access with dynamic access barring for mtc in 3gpp lte-a networks. *Vehicular Technology, IEEE Transactions on*, 63(5):2467–2472.

Lo, A., Law, Y. W., Jacobsson, M., and Kucharzak, M. (2011). Enhanced lte-advanced random-access mechanism for massive machine-to-machine (m2m) communications. In *27th World Wireless Research Forum (WWRF) Meeting*, pages 1–5.

NS-3 (2014). *The network simulator ns-3*.

O'Neill, B. (1982). A problem of rights arbitration from the talmud. *Mathematical Social Sciences*, 2(4):345–371.

Osborne, M. J. and Rubinstein, A. (1994). *A course in game theory*. MIT press.

Pang, Y.-C., Chao, S.-L., Lin, G.-Y., and Wei, H.-Y. (2014). Network access for m2m/h2h hybrid systems: a game theoretic approach. *Communications Letters, IEEE*, 18(5):845–848.

Shapley, L. S. (1952). A value for n-person games. Technical report, DTIC Document.

ZTE (2011). Backoff enhancements for ran overload control. Technical report, 3GPP.

Mapeamento de Redes Virtuais Secundárias em Substrato Sem Fio Baseado em Rádio Cognitivo: Formulação Multi-objetivo e Análise

Andson M. Balieiro[1,2], Kelvin L. Dias[1]

[1]Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
50670-901 – Recife – PE – Brasil

[2]Universidade de Pernambuco (UPE)- Campus Garanhuns

{amb4, kld}@cin.ufpe.br

Abstract. *Wireless Virtualization (WV) is put forward as a solution to manage the heterogeneous wireless communication environment. Although it provides better resource utilization, the underutilization of resources can still occur when the current approaches are adopted. This problem can be overcome by combining WV with the cognitive radio (CR) technology and dynamic access spectrum (DSA) techniques. Thus, Primary Virtual Networks (PVNs) and Secondary Virtual Networks (SVNs) can be overlaid, where the SVNs access the resources opportunistically. This paper presents the cognitive radio virtual networks environment. It formulates the problem of SVN mapping onto substrate network based on CR and analyzes the related metrics.*

Resumo. *A virtualização sem fio (WV) é vista como uma solução para gerenciar o ambiente de comunicação sem fio heterogêneo. Embora ela proporcione melhor utilização de recursos, a subutilização deles ainda pode ocorrer utilizando as abordagens atuais. Isto pode ser superado através da combinação de WV com rádio cognitivo (CR) e técnicas de acesso dinâmico ao espectro (DSA). Assim, redes virtuais primárias (PVNs) e secundárias (SVNs) podem ser sobrepostas, onde as redes SVNs utilizam os recursos oportunistamente. Este artigo apresenta o ambiente de redes virtuais de rádio cognitivo, formula o problema de mapeamento de redes virtuais secundárias (SVN) em redes de substrato (SN) baseadas em CR e analisa as métricas envolvidas.*

1. Introdução

As tecnologias de comunicação móvel estão progredindo rapidamente tal que uma densa e heterogênea rede sem fio é vislumbrada. A demanda por serviços sem fio com diferentes requisitos tem crescido ultimamente, e gerenciar este complexo ecossistema é um desafio. A virtualização sem fio (WV) surge como uma solução eficiente, visto que redes sem fio virtuais (VWNs) de diferentes requisitos podem ser criadas, compartilhando a mesma infraestrutura sem fio e fornecendo serviços diferenciados aos usuários. WV envolve o compartilhamento de infraestrutura e espectro, e introduz novos atores ao modelo de negócio: o operador de rede móvel (MNO) e o provedor de serviço (SP) [Liang 2014].

Para satisfazer a grande demanda por comunicação móvel, o espectro eletromagnético deve ser feito disponível [Akyildiz 2006]. Embora a WV proporcione melhor uso dos recursos, as abordagens atuais para a WV ainda podem causar subutilização deles. Nestas abordagens, o recurso é alocado de forma exclusiva a uma VWN durante o tempo de vida dela. Devido à variação de carga de tráfego nas redes, podem existir casos onde as VWNs não estejam utilizando todos os seus recursos alocados. Esta subutilização pode ter efeito adverso na implantação de novas VWNs e conduzir a perda de receita para o provedor de infraestrutura.

Combinar WV com CR e técnicas de DSA [Akyildiz 2006] pode superar este problema, onde novas VWNs podem ser implantadas através do compartilhamento oportunista de recursos e a virtualização baseada em espectro pode ser alcançada [Liang 2014]. Isto permite ter um ambiente composto por VWNs com diferentes prioridades de acesso ao recurso, Redes Virtuais Primárias (PVNs) e Redes Virtuais Secundárias (SVNs), que são implantadas de forma sobreposta e compartilham a mesma rede de substrato, onde a SVN somente tem acesso ao recurso quando a PVN não o está utilizando. Através da cognição e reconfigurabilidade, o CR possibilita a implantação deste ambiente. Entretanto, novos desafios emergem, desde o mapeamento até a operação dessas redes.

Mapeamento de redes sem fio virtuais em rede de substrato é um problema NP - difícil [Belbekkouche 2012] e envolve reservar e alocar recursos físicos aos elementos que compõe as redes virtuais tais como estações base (BSs) virtuais e canais de comunicação virtuais. Este problema torna-se mais desafiador quando se considera um ambiente formado por PVNs e SVNs, onde o mapeamento de SVN deve considerar não apenas a demanda solicitada em termos de número de usuários (chamados de usuários secundários virtuais - VSUs), por exemplo, de modo a prover bons níveis de QoS aos VSUs, mas também o padrão de uso dos recursos pelos usuários das PVNs (usuários primários virtuais - VPUs) a fim de não causar interferência na comunicação das PVNs.

Desta forma, o mapeamento de SVNs impõe restrições e objetivos que estão relacionadas à comunicação das PVNs e SVNs, bem como ao provedor de infraestrutura, que busca garantir utilização eficiente de recursos. Este mapeamento envolve vários objetivos e uma formulação que leva em conta tal aspecto precisa ser realizada. Além disso, é importante analisar as métricas relacionadas aos objetivos a fim de entender quais parâmetros impactam neles, bem como quais objetivos são conflitantes entre si. Para o melhor do nosso conhecimento, este é o primeiro estudo a: formular o problema de mapeamento de SVNs em redes de substrato como um problema multi-objetivo; definir o ambiente de redes virtuais de rádio cognitivo; analisar a influência de importantes parâmetros/métricas sobre outras métricas de modo a prover uma valiosa assistência no projeto de esquemas para resolver o problema proposto.

Este artigo está assim organizado. A Seção 2 apresenta os trabalhos relacionados. O ambiente de redes virtuais de rádio cognitivo é descrito na Seção 3. Métricas e o problema multi-objetivo de mapeamento de SVNs em rede de substrato são formuladas na Seção 4. Seção 5 conduz uma análise de influência de parâmetros/métricas sobre outras. Seção 6 conclui este artigo.

2. Trabalhos Relacionados

Vários estudos têm sido propostos para virtualização de redes em ambos os ambientes cabeados [Zhang 2012] e sem fio [Zaki 2010]. Em relação WV, alguns trabalhos adotam: uma tecnologia específica [Zaki 2010][Banchs 2012]; uma abordagem genérica, sem especificar qualquer tecnologia adotada na rede de substrato [Fu 2010]; ou cenários com redes sem fio heterogêneas [Caeiro 2012]. Estes estudos consideram que o recurso alocado a uma rede virtual não pode ser alocado a outra, mesmo se a primeira não o está utilizando. Esta restrição pode causar a subutilização de recursos em períodos de baixa carga de tráfego.

Nesta direção, em [Zhang 2012] o problema de mapeamento de redes virtuais em redes de substrato cabeadas, baseado no compartilhamento oportunista de recursos, é apresentado. Os autores consideram a carga de trabalho das redes virtuais formada por uma subcarga básica, que sempre existe, e uma variável, que ocorre com uma dada probabilidade. Assim, múltiplos tráfegos variáveis de diferentes redes virtuais compartilham algum recurso em comum de modo a alcançar uma melhor utilização de recursos. Entretanto, este compartilhamento pode ocasionar colisão/interferência entre tráfegos variáveis de diferentes redes virtuais.

Em [Yang 2013] os autores transferem a formulação de [Zhang 2012] para o ambiente sem fio. Como em [Zhang 2012], em [Yang 2013] os autores consideram os recursos como sendo homogêneos em termos de largura de banda oferecida. Entretanto, isto não engloba cenários com redes heterogêneas (ex. WiMax, WiFi e LTE), onde diferentes tecnologias possuem unidades de recursos distintos e diferentes taxas de dados [Wang 2013]. Além disso, mesmo com uma única tecnologia de rede, fator tal como o ruído pode impactar na taxa de dados alcançada em um canal de comunicação [Chena 2011]. Diferente de nossa proposta, em [Zhang 2012] e [Yang 2013], as redes virtuais possuem o mesmo nível, direito de acesso ao recurso, com nenhuma distinção entre rede/usuário virtual primária (maior prioridade) e rede/usuário virtual secundária (menor prioridade). Em vista disso, elas não incluem estes dois elementos do ambiente de CR: as redes/usuários secundárias e a primárias. Além disso, em cenários formados por redes com diferentes prioridades de acesso aos recursos, existem outros fatores que afetam a comunicação dos usuários e que devem ser considerados no mapeamento de redes virtuais, tais como as probabilidades de bloqueio de usuários secundários e *handover* de usuários secundários, que são negligenciados em ambos os estudos.

Uma plataforma de virtualização de rede fim-a-fim é proposta em [Nakauchi 2011]. Ela busca criar redes virtuais com recursos cabeados e sem fio e usa CR para gerenciar a WV. Entretanto, os autores abordam somente a conexão transparente das redes de acesso sem fio com a rede cabeada. Eles não formulam o problema de mapeamento de redes virtuais em rede de substrato, nem analisam as importantes métricas/objetivos envolvidas.

Em [Xin 2012] os autores adotam uma abordagem denotada como acesso ao espectro dinâmico como um serviço para alcançar DSA. Esta abordagem dinamicamente oferece serviço de espectro aos usuários. Eles adotam alocação de espectro dinâmica para DSA, que não distingue entre PUs e SUs. Assim, cada usuário tem uma banda de espectro para uso exclusivo durante um período. Este tipo de alocação pode causar subutilização de espectro, quando a carga de tráfego é baixa. Além disso, a artigo

considera somente demandas homogêneas (todas as redes virtuais possuindo a mesma demanda espectro), o que não é sempre verdade em cenários reais.

Diferente de [Xin 2012], nosso trabalho adota a abordagem de acesso oportunista ao espectro (OSA) para DSA, onde a diferença entre usuários primários e secundários é estabelecida. Em OSA, os SUs buscam e acessam dinamicamente as bandas de espectro de forma oportunista através do sensoriamento de espectro ou banco de dados de espectro [Min 2011]. Neste aspecto, no mapeamento de SVNs, consideramos a existência de VPNs, que tem recursos alocados as SVNs e maior prioridade de acesso a eles. Além disso, demandas heterogêneas são levadas em conta na nossa formulação.

Nota-se na literatura que o mapeamento de redes fio virtuais tem sido bastante estudado, bem como as formas de acesso oportunistas em redes não virtualizadas. Porém, quando se considera esses dois elementos no mapeamento de redes virtuais, não há trabalho com formulação similar a apresentada neste artigo.

3. O Ambiente de Redes Virtuais de Rádio Cognitivo

O ambiente de redes virtuais de radio cognitivo é formado por três tipos de redes sem fio: rede de substrato/infraestrutura, redes primárias virtuais (PVNs) e redes secundárias virtuais (SVNs). Elas estão representadas na Fig.1, com cada tipo em uma camada específica. A camada física engloba as redes de substrato, que consiste de canais/bandas de espectro, estações-base, servidores e outros equipamentos que compõe a infraestrutura do ambiente sem fio. As redes de substrato são usadas na instanciação de ambas as redes virtuais (PVNs e SVNs). Em arquiteturas como a proposta em [Nakauchi 2011], o provedor de infraestrutura é responsável por gerenciar os recursos físicos.

As PVNs têm maior prioridade de acesso aos recursos do que as SVNs. O mapeamento de PVN é realizado sem considerar a existência de SVNs, de forma usual [Fu 2010]. Assim, os recursos são alocados para cada PVNs exclusivamente. Como a carga de tráfego das PVNs varia ao longo do tempo, podem existir casos onde as PVNs não estão utilizando todos os seus recursos. Diante desses períodos, as SVNs podem ser implantadas para utilizar os recursos subutilizados oportunistamente.

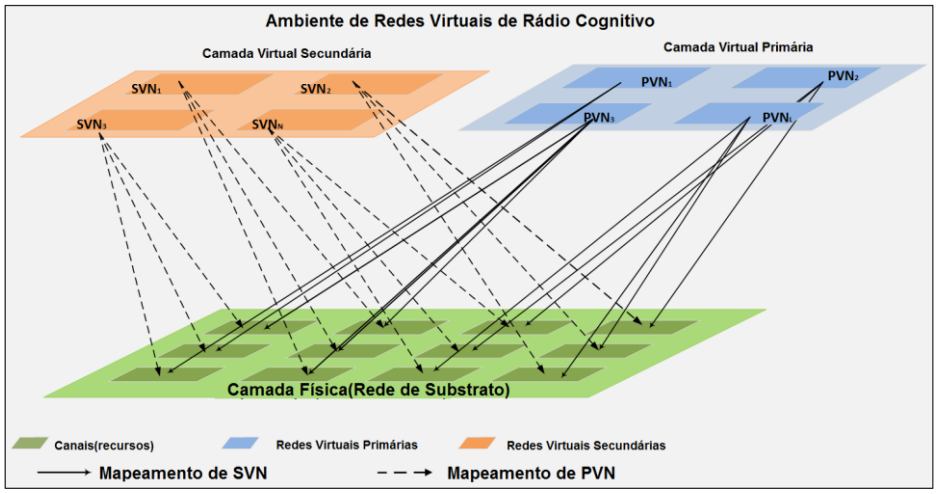


Figura 1 Ambiente de redes virtuais de rádio cognitivo

A adoção das SVNs pode proporcionar melhor utilização de recursos (ex. espectro) e aumentar a receita do provedor de infraestrutura, uma vez que mais redes virtuais podem ser admitidas. Entretanto, para prover acesso oportunista, os usuários da SVNs devem identificar a atividade dos VPUs e somente acessar os recursos na ausência deles. Neste aspecto, o CR é uma tecnologia essencial para a implantação das SVNs, devido as suas capacidades de cognição e reconfiguração [Akyildiz 2006]. Em um ambiente formado por PVNs e SVNs, o mapeamento das redes virtuais torna-se mais complexo, uma vez que o mapeamento de SVNs deve considerar tanto a demanda requisitada pela SVN quanto a atividade dos usuários das PVNs.

4. Modelagem do Mapeamento de SVNs em Redes de Substrato Sem Fio

Em um ambiente de redes virtuais de rádio cognitivo, o provedor de serviço solicita e gerencia L PVNs [Liang 2014]. Dado que a rede de substrato é composta por M canais e que os recursos são divididos entre as PVNs segundo a porcentagem q_j, com $j = 1,2,3,...,L$, $0 \le q_j \le 1$, $\sum_{j=1}^{L} q_j = 1$ e Q_j sendo o conjunto de canais alocados a PVN j, com $|Q_j| = \lfloor M.q_j \rfloor \; ou \; \lceil M.q_j \rceil$ canais, onde, $\lceil x \rceil$ e $\lfloor x \rfloor$ são as funções piso e teto, respectivamente. Dado que a chegada de VPUs no canal $i\,(C_i)$ da rede virtual j, com $C_i \in Q_j$, segue um processo de Poisson com taxa de chegada $\lambda_{i,j}^{PU}$, e o tempo de permanência do usuário é dado por uma distribuição exponencial com média $\frac{1}{\mu_{i,j}^{PU}}$, a utilização do canal i é dada pela dada pela Eq. 1 [Bolch 2006]. Um ambiente com acesso oportunista (pelos VSUs) aos canais é obtido pela adoção do critério de estabilidade $\rho_{i,j}^{PU} < 1$.

$$\rho_{i,j}^{PU} = \left(\frac{\lambda_{i,j}^{PU}}{\mu_{i,j}^{PU}} \right) \tag{1}$$

Nesta abordagem, similarmente a [Akter 2008], nós abstraímos a existência de vários VPUS no mesmo canal. Assim, cada canal é ocupado por um VPU por vez. Esta consideração é válida em sistemas cujo mecanismo de acesso é baseado em divisão de freqüência/tempo, por exemplo. Assim, a probabilidade do canal estar ocupado pelo VPU (em estado ON) é dada pela sua utilização (Eq.1). A probabilidade do canal não estar ocupado pelo VPU (estado OFF) é dada pela Eq. 2.

$$POFF_{i,j} = (1 - \rho_{i,j}^{PU}) \tag{2}$$

O número médio de canais em estado OFF na rede virtual j é dado pela Eq.3.

$$E[NCh_j^{OFF}] = \sum_{n=1}^{|Q_j|} n.P[NCh_j^{OFF} = n] \tag{3}$$

Onde $P[NCh_j^{OFF} = n]$ é a probabilidade de n canais estarem OFF na rede virtual j. Sendo $A_{d,n,j}$ a partição d com n elementos do conjunto Q_j, com $d = 1,2,3...,s$,

onde s é o número de partições com n elementos do conjunto Q_j, dado por $s = \begin{pmatrix} |Q_j| \\ n \end{pmatrix}$, o valor de $P[NCh_j^{OFF} = n]$ pode ser dado pela Eq.4. De forma similar, a probabilidade de que existam n VPUs na rede virtual j é dada pela Eq. 5.

$$P[NCh_j^{OFF} = n] = \sum_{d=1}^{s} \left[\prod_{i \in A_{d,n,j}} POFF_{i,j} . \prod_{h \in \{Q_j - A_{d,n,j}\}} [1 - POFF_{h,j}] \right] \quad (4)$$

$$P[NPU_j = n] = \sum_{d=1}^{s} \left[\prod_{i \in A_{d,n,j}} \rho_{i,j}^{PU} . \prod_{h \in \{Q_j - A_{d,n,j}\}} [POFF_{h,j}] \right] \quad (5)$$

O ambiente de comunicação sem fio é influenciado por fatores, tais como ruídos provenientes de outras fontes. Estes fatores afetam a força do sinal percebido no receptor. Assim, para denotar as condições do canal primário, considera-se que a relação sinal ruído no canal i da rede virtual j, $SNR(db)_{i,j}$, é dada por uma distribuição exponencial com média $\frac{1}{\lambda_{i,j}^{SNR(db)}}$ dB [Choudhury 2007]. Assim, a capacidade máxima (em bps) do canal i alocado a rede virtual j, denotada como $R_{i,j}$, dada pela Lei de Shannon [Shannon 2011], é expressa na Eq.6, onde $Bw_{i,j}$ é a largura de banda do canal (em Hertz).

$$R_{i,j} = Bw_{i,j} . \log_2(1 + SNR_{i,j}) \quad (6)$$

O valor da $SNR_{i,j}$ usando escala decimal obtida daquela em decibéis (dB) é dada por $SNR_{i,j} = 10^{\frac{SNR(db)_{i,j}}{10}}$, e a capacidade media alcançada no canal i da rede virtual j ($\overline{R_{i,j}}$) é dada pela Eq. 7.

$$\overline{R_{i,j}} = E[R_{i,j}] = E[Bw_{i,j} . \log_2(1 + (SNR_{i,j}))] \quad (7)$$

A Eq.7 representa a capacidade média do canal quando ele está sempre disponível para uso pelo usuário. Quando o usuário é o VSU, isto não ocorre devido ao uso oportunista. Assim, dada a Eq.8 e a probabilidade do canal estar em estado OFF (Eq.2), a taxa efetiva média alcançada pelo VSU é dada pela Eq. 8.

$$\overline{Re_{i,j}} = POFF_{i,j} . \overline{R_{i,j}} \quad (8)$$

No mapeamento de redes virtuais, não há um relacionamento um-para-um entre PVNs e SVNs e vice-versa. Assim, canais alocados a diferentes PVNs podem ser usados pela mesma SVN, como mostrado na Fig. 1, onde a SVN #3 utiliza os canais alocados as PVN #1 e PVN #3, por exemplo, o que fornece maior possibilidade de mapeamento para as SVNs. Isto é diferente de [Lai 2011] e [Wang 2009], que adotam uma abordagem que restringe ao relacionamento unívoco entre PVNs e SVNs. Deste modo, adota-se o nível de granularidade de canal é adotado para informação da atividade primária.

Consideram-se existem N SVNs a serem mapeadas. Para cada *SVN* l (SVN_l), a largura de banda requisitada pelos VSUs é dada por uma distribuição exponencial com média $1/w_l$ bps. A chegada de VSUs na SVN_l segue um processo de Poisson com taxa de λ_l^{SU} usuários por segundos. O tempo de serviço do VSU é exponencialmente distribuído com média $1/\mu_l^{SU}$ segundos [Zhao 2011]. Assim, o número médio de VSUs na SVN_l, e o total de recursos requisitados são dados pelas Eq. 9 e Eq.10.

$$\overline{NSU_l} = \lambda_l^{SU} . \frac{1}{\mu_l^{SU}} \tag{9}$$

$$\overline{Bw_{req,l}} = \lambda_l^{SU} . \frac{1}{\mu_l^{SU}} . \frac{1}{w_l} = \overline{NSU_l} . \frac{1}{w_l} \tag{10}$$

4.1. Formulação para a Probabilidade de Colisão

No mapeamento de SVNs, como os canais adotados são compartilhados com as PVNs, é necessário garantir que o nível de interferência causada na PVNs não ultrapasse um limite definido. Este limite pode ser baseado em acordo/especificação em nível de serviço (SLA/SLS) das PVNs ou no nível de interferência que pode ser tolerado pelo sinal do VPU, por exemplo. Assim, na seleção dos canais alocados a cada SVN, a interferência/colisão entre VPU e VSU deve ser calculada para garantir que não ultrapasse o limiar definido.

Dado que o mapeamento da SVN_l adotou o conjunto de canais $SC_l = \{C_1, C_2, ..., C_n\}$, onde $|SC_l| = n_l$, $SC_l \subset \bigcup_{j=1}^{L} Q_j$, e $SC_l \cap SC_u = \varnothing$, para todo $l \neq u$, com $l, u = 1, 2, 3, ..., N$ sendo identificadores de SVNs. No mapeamento de redes virtuais (secundária ou primária), quando cada usuário necessita de somente um canal para realizar a sua comunicação, uma colisão entre VPU e VSU na SVN_l pode ocorrer quando o número de VPUs e VSUs que tentam acessar os canais simultaneamente é maior que o número de canais alocados a SVN_l, com pelo menos um VPU e um VSU entre os usuários. Assim, a probabilidade de colisão é dada pela Eq.11.

$$Pc_l(SC_l) = \sum_{i=1}^{n_l} (P[NPU_l = i \; ; \; NSU_l > n_l - i]) \tag{11}$$

Onde $P[NPU_l = i \; ; \; NSU_l > n_l - i]$ é a probabilidade que no conjunto de canais SC_l, o número de VPUs acessando ou tentando acessar os canais é i o número de VSUs é maior do que $n_l - i$. Como a chegada e tentativa de acesso dos usuários ocorre independente de cada outro, $P[NPU_l = i \; ; \; NSU_l > n_l - i]$ pode ser obtida do produto entre $P[NPU_l = i]$ e $P[NSU_l > n_l - i]$, que são dadas pelas Eq.5 e Eq.12.

$$P[NSU_l > n_l - i] = 1 - \left(\sum_{k=0}^{n_l - 1} P[NSU_l = k] \right) \tag{12}$$

Onde

$$P[NSU_l = k] = \frac{\left(\lambda_l^{SU} \cdot \frac{1}{\mu_l^{SU}}\right)^k e^{-\left(\lambda_l^{SU} \cdot \frac{1}{\mu_l^{SU}}\right)}}{k!} \tag{13}$$

Entretanto, nesta formulação, a condição de que cada usuário necessita somente de um canal para atender sua demanda não se aplica ao VSU, apenas ao VPU. Como a largura de banda requisitada pelo VSU pode ser maior do que a capacidade média de um canal, quando o VSU tenta acessar mais do que um canal na SVN, ele pode colidir com mais do que um VPU. Dessa forma, a probabilidade de colisão é dada pela Eq.14.

$$Pc_l(SC_l) = \sum_{i=1}^{n}\left(P\left[NPU_l = i \; ; NSU_l > \left\lfloor \frac{n_l - i}{ChSU_l} \right\rfloor\right]\right) \tag{14}$$

Onde $ChSU_l$ é o número médio de canais requisitado para atender a demanda de cada VSU na SVN_l, que é dado pela Eq.15, $\lfloor . \rfloor$ é a função piso e $\overline{Rch_l}$ é a capacidade média efetiva dos canais da SVN_l dada pela Eq.16. A probabilidade de colisão média com N SVNs mapeadas é dada pela Eq. 17.

$$ChSU_l = \max\left(1, \frac{1/w_l}{\overline{Rch_l}}\right) \tag{15}$$

$$\overline{Rch_l} = \frac{\sum_{i=1}^{n_l} \overline{Re_{i,l}}}{n_l} \tag{16}$$

$$\overline{Pc} = \frac{\sum_{l=1}^{N} Pc_l}{N} \tag{17}$$

4.2. Formulação para a Probabilidade de Bloqueio

Assim como restringir a colisão no VPU, o mapeamento de SVNs deve prover bons níveis de QoS para o VSU. Ele deve admitir o máximo possível de VSUs dimensionados para cada SVN. Assim, a probabilidade de bloqueio de VSU deve ser determinada no processo de mapeamento. O bloqueio de VSU ocorre na SVN_l quando o número total de VPUs com o número médio de canais requisitados pelos VSUs é maior do que o número de canais alocados a SVN_l e sua probabilidade é dada pela Eq. 18.

$$Pb_l^{SU}(SC_l) = \sum_{i=0}^{n}\left(P[NPU_l = i \; ; NSU_l > \left\lfloor \frac{n_l - i}{ChSU_l} \right\rfloor]\right) \tag{18}$$

Com N SVNs mapeadas, a probabilidade bloqueio média é dada na Eq.19.

$$\overline{Pb^{SU}} = \frac{\sum_{l}^{N} Pb_l^{SU}}{N} \tag{19}$$

4.3. Formulação para a Utilização Conjunta

Prover melhor utilização de recurso é também um objetivo do mapeamento de SVN. Para definir a utilização de recursos alcançada no mapeamento, as atividades das PVNs e SVNs devem ser consideradas. Assim, dado o conjunto de n_l canais adotado no mapeamento da SVN_l, a utilização conjunta (primária e secundária) dos canais é dada pela Eq.20. Onde $\overline{NPU_l}$ é o número médio de VPUs nos canais, dado pela Eq. 21.

$$util_l = \frac{\overline{NPU_l} + (1 - Pb_l^{SU}).\overline{NSU_l}.ChSU_l}{n_l} \tag{20}$$

$$\overline{NPU_l} = n_l - E[NCh_l^{OFF}] \tag{21}$$

O produto $(1 - Pb_l^{SU}) * \overline{NSU_l}$ denota o número médio de VSUs admitidos na SVN_l, onde $\overline{NSU_l}$ é dado pela Eq. 9, e Pb_l^{SU} pela Eq. 18. Com N SVNs mapeadas, a utilização conjunta é dada pela Eq.22.

$$\overline{util} = \frac{\sum_{l=1}^{N} util_l}{N} \tag{22}$$

4.4. Formulação para a Probabilidade de *Handover* de SVN

A probabilidade de bloqueio de VSU indica o nível de rejeição de novos VSUs na SVN. Uma vez que eles são admitidos, alguns eventos desecandeados pela atividade primária podem afetar as suas comunicações. Entre estes eventos, o *handover* de SVN é destacado. Ele acontece quando o VSU vaga canal que ocupa devido ao retorno do VPU e seleciona um canal em outra SVN para retomar a sua comunicação, visto que não existe canal disponível na sua rede atual. O handover de SVN causa degradação na comunicação secundária [Lai 2011].

A probabilidade de handover do VSU oriundo da SVN_l pode ser obtida pela razão entre o número de tentativas de *handover* com sucesso (aqueles que encontraram recursos disponíveis em outra SVN) e o número de VSU admitidos na SVN_l. O número de VSUs admitidos na SVN_l (NAd_l^{SU}) é dado pela Eq. 23. O número de tentativas de *handover* de SVN (NHA_l) é dada pela Eq.24.

$$NAd_l^{SU} = (1 - Pb_l^{SU})\lambda_l^{SU}.\frac{1}{\mu_l^{SU}} \tag{23}$$

$$NHA_l(SC_l) = PcAd_l.NAd_l^{SU} = PcAd_l.(1 - Pb_l^{SU})\lambda_l^{SU}.\frac{1}{\mu_l^{SU}} \tag{24}$$

Onde $PcAd_l$ é a probabilidade de colisão entre os VSUs admitidos na SVN_l e os VPUs, que pode ser obtida de forma similar a Eq. 14, onde $P[NSU_l = k]$, dada na Eq.13, é alterada para a Eq. 25.

$$P[NSU_l = k] = \frac{\left((1 - Pb_l^{SU}).\lambda_l^{SU} . \frac{1}{\mu_l^{SU}}\right)^k e^{-\left((1-Pb_l^{SU}).\lambda_l^{SU} . \frac{1}{\mu_l^{SU}}\right)}}{k!} \quad (25)$$

A probabilidade de tentativa de *handover* dos VSUs da SVN_l é dada pela Eq.26.

$$P_{HA,l}(SC_l) = \frac{NHA_l}{NAd_l^{SU}} = PcAd_l \quad (26)$$

O número de número de *handover*s com sucesso depende da quantidade de recursos disponíveis nas outras redes. Dado que existam (em média) R recursos disponíveis, o número de *handovers* de SVN com sucesso é dado pela Eq. 27.

$$NH_l = \frac{Min\ (NHA_l\ .\ ChSU_l*, R)}{ChSU_l*} \quad (27)$$

Onde $ChSU_l*$ é o número médio de canais necessários para continuar a comunicação de cada VSU que tenta realizar o *handover* da SVN_l, obtido de forma similar a Eq. 15, considerando a capacidade média efetiva dos canais alocados as outras SVNs (exceto a SVN_l) como denominador. O valor de R é dado pela Eq. 28.

$$R = \max\left(0, \sum_{j=1, j\neq l}^{N} \left[n_j - \left(\overline{NPU}_j + (1 - Pb_j^{SU}).\ \overline{NSU}_j\ .\ ChSU_j\right)\right]\right) \quad (28)$$

Considerando os VSUs admitidos na SVN_l, a probabilidade de *handover* de SVN é dada pela Eq. 29. Com N SVNs, a probabilidade média de *handover* de SVN é dada pela Eq. 30.

$$P_{handover,l} = \frac{NH_l}{NAd_l^{SU}} \quad (29)$$

$$\overline{P_{handover}} = \frac{\sum_{l=1}^{N} P_{handover,l}}{N} \quad (30)$$

4.5. O Problema Multi-Objetivo de Mapeamento de SVNs

Como apresentado nas seções anteriores, vários objetivos estão envolvidos no mapeamento de SVNs. Quando a influência de cada um deles é considerada, o seguinte problema de otimização pode ser formulado (ver Eq. 31). Dado um conjunto de N SVNs e o padrão de uso primário dos canais, realizar o mapeamento das SVNs em substrato físico de modo a minimizar o handover entre SVNs e o bloqueio de VSUs e maximizar a utilização conjunta. Três restrições devem ser satisfeitas: a probabilidade de colisão deve estar abaixo de um dado limiar, o montante de recursos alocados a cada SVN não pode ser menor do que o requisitado e um recurso comum não pode ser alocado à diferentes SVNs. Em notação, tem-se:

$$\textit{Minimize } \overline{P_{handover}},\ \overline{Pb^{SU}} \text{ and } \textit{Maximize } \overline{util}$$

Sujeito a :

$$\begin{cases} Pc_l < thr_{collision},\ l = 1,2,3,...,N \\ Ralloc_l \geq \overline{Bw_{req,l}},\ l = 1,2,3,...,N \\ SC_l \cap SC_u = \varnothing,\ l \neq u,\ l,u = 1,2,3,...,N \end{cases} \quad (31)$$

5. Análise do Mapeamento de Redes Virtuais Secundárias

As restrições e objetivos no problema de mapeamento de SVN descritos na seção anterior, como a demanda requisitada pela SVN, o limite de colisão, redução do bloqueio de VSU e *handover* de SVN e maximização da utilização conjunta de recursos estão relacionados à comunicação primária, secundária ou ao provedor de infraestrutura. Alcançar todos estes objetivos simultaneamente é um processo desafiador, porque alguns são conflitantes entre si e o mapeamento com foco em um pode deteriorar os outros. Nesta seção a análise de métricas relacionadas aos objetivos é conduzida através da variação de importantes parâmetros.

A utilização primária dos canais é um importante parâmetro que deve ser considerado no mapeamento de SVN, uma vez que ele afeta diretamente métricas como as probabilidades de colisão, bloqueio de VSU e utilização conjunta.

A Fig. 2 considera o número de canais alocados a SVN, a taxa de chegada e tempo médio de serviço do VSU fixos e mostra o comportamento das probabilidades de colisão, bloqueio de VSU e utilização conjunta quando a utilização primária dos canais alocados a SVN varia. Baixas probabilidades de bloqueio de VSU e colisão podem ser alcançadas selecionando canais com baixa utilização primária para mapear a SVN. Entretanto, selecionar canais com baixa utilização primária pode reduzir a utilização conjunta de recursos, mesmo que uma alta taxa de admissão de VSUs seja alcançada, que é inferida pela baixa probabilidade de bloqueio de VSU e alta utilização secundária de recursos. Por outro lado, quando canais com alta utilização primária são selecionados, melhor utilização conjunta é alcançada (ver Fig. 2). Mas, isto pode causar um aumento no bloqueio de VSUs e na colisão, que afeta as PVNs e SVNs.

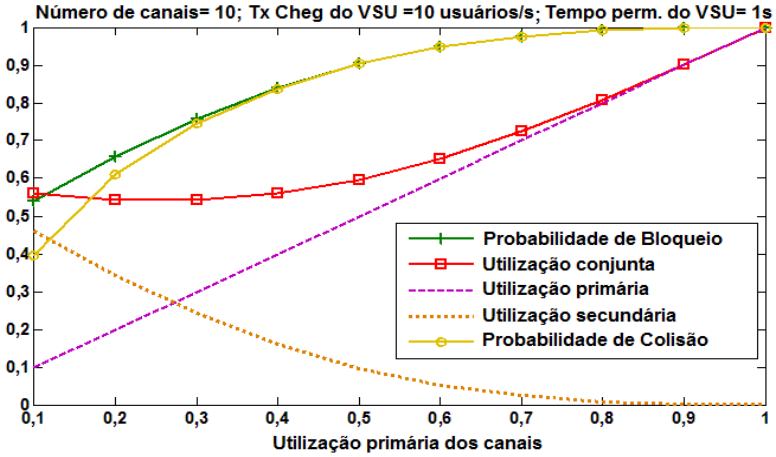


Figura 2. Influência da utilização primária no mapeamento de SVN

Embora exista um aumento na utilização conjunta quando canais com alta utilização primária são selecionados, a utilização provida pelos VSUs decresce até alcançar um valor nulo, o que significa que não há possibilidade de acesso oportunista (ver Fig. 2). Assim, no mapeamento de SVN é importante balancear a seleção de canais para alcançar um bom compromisso entre a colisão, bloqueio e utilização conjunta.

A Fig. 3 ilustra os resultados das probabilidades de colisão, bloqueio de VSU e tentativa de *handover* e utilização conjunta, quando o número de canais adotados no mapeamento da SVN varia e a utilização primária, taxa de chegada, tempo médio de serviço do VSU são fixos. Nota-se que as probabilidades de colisão e bloqueio de VSU decrescem quando o núlmero de canais alocados aumenta. Isto indica que existe menor interferência na comunicação do VPU e poucos VSUs são rejeitados na SVN, que são fatores positivos. Além disso, com mais canais, em geral, existem mais recursos para atender a demanda dos VSUs, o que reduz a chance de acontecer *handover* de SVN. Entretanto, aumentar o número de canais a partir de um dado ponto (14 canais), usualmente conduz a uma redução na utilização conjunta, como ilustrado na Fig. 3, porque, apesar da baixa taxa de bloqueio de VSU na SVN, o montante de recursos alocados a SVN é muito maior do que a demanda a ser atendida. Por outro lado, caso um número reduzido seja adotado, a utilização conjunta aumenta, o que é um fator positivo. Entretanto, a colisão e o bloqueio de VSU também aumentam, o que é um fator negativo.

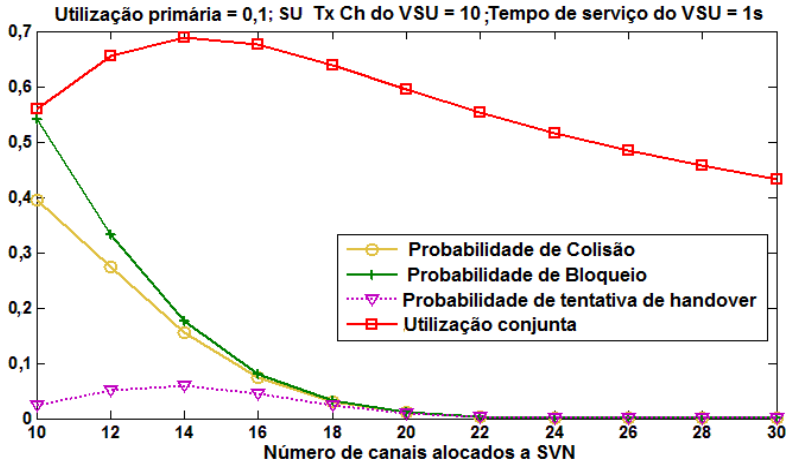


Figura 3. Análise da variação do número de canais alocados a SVN

É importante notar (Fig. 3) o comportamento da tentativa de *handover* de SVN (que compõe a probabilidade de *handover* de SVN) quando o número de canais varia em uma dada faixa (entre 10 e 14). Nesta faixa, quando o número de canais aumenta, a utilização conjunta e a probabilidade de tentativa de *handover* também aumentam. Na utilização conjunta, com mais canais, mais VSUs são admitidos e o numerador da Eq. 20 cresceu mais do que o denominador. A maior admissão de VSUs aumentou a taxa de ocupação da SVN, ocasionando maior possibilidade do VSU comutar de SVN.

Nota-se que probabilidade de bloqueio do VSU tem um impacto na tentativa de *handover* e utilização conjunta. Assim, a Fig.4 ilustra o comportamento destas duas métricas quando diferentes valores para a probabilidade de bloqueio de VSU são definidos. O aumento do bloqueio de VSU provoca redução na utilização conjunta e na

tentativa de *handover*, pois poucos VSUs são admitidos na SVN. Dessa forma, a influência do bloqueio deve ser considerada no processo de mapeamento de SVNs.

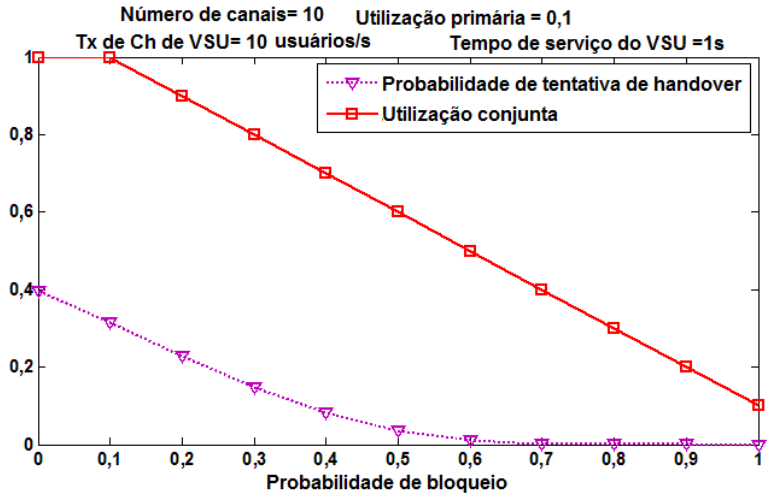


Figura 4. Influência da probabilidade de bloqueio

6. Conclusão e Direções Futuras

Neste artigo destacamos a combinação de rádio cognitivo, técnicas de DAS e virtualização sem fio no mapeamento de redes sem fio virtuais, possibilitando que redes virtuais com diferentes prioridades de acesso ao recurso coexistam. Vimos que mapeamento de SVNs é um problema desafiador com objetivos relacionados ao VPU, VSU e provedor de infraestrutura, onde uma análise de influência de parâmetros/métricas sobre métricas/objetivos foi conduzida. Ela buscou prover subsídios para a elaboração de esquemas para resolver o problema proposto.

Trabalhos futuros incluem o desenvolvimento de esquema para resolver o problema proposto (ex. baseados em inteligência artificial) e a avaliá-lo em termos das métricas aqui formuladas.

Referências

Akter, L., Natarajan, B., Scoglio, C.(2008) “Modeling and Forecasting Secondary User Activity in Cognitive Radio Networks”, Proceedings of 17th International Conference on Computer Comm. and Networks (ICCCN), pp. 1-6.

Akyildiz, I. F. et al. (2006) “Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey”, Elsevier Computer Networks (50).

Banchs, A. et al. (2012) “Providing Throughput and Fairness Guarantees in Virtualized WLANs Through Control Theory”, Springer Mobile Networks and Applications, vol.17, issue 4, pp 435-446.

Belbekkouche, A., Hasan, M. M., Karmouch, A.(2012) “Resource Discovery and Allocation in Network Virtualization”, IEEE Communications Surveys & Tutorials, Vol.14, Issue 4, pp.1114-1128.

Bolch, G. et al. (2006) “Queueing Networks and Markov Chains”, Second Edition, Wesley.

Caeiro, L., Cardoso, F. D., Correia , L. M.(2012) “Adaptive allocation of Virtual Radio Resources over heterogeneous wireless networks”, 18th European Wireless Conference (European Wireless).

Chena , Y-S. et al. (2011) “A Cross-Layer Protocol of Spectrum Mobility and Handover in Cognitive LTE Networks”, Elsevier Simulation Modelling Practice and Theory, vol. 19, issue 8, pp.1723–1744.

Choudhury, S., Gibson, J. D. (2007) “Information Transmission Over Fading Channels”, IEEE Global Telecom. Conf. (GLOBECOM), pp. 3316-3321.

Fu, F., Kozat, U. C. (2010) “Wireless Network Virtualization as A Sequential Auction Game”,Proceedings of IEEE INFOCOM.

Lai, J. et al. (2011) “Network Selection in Cooperative Cognitive Radio Networks”, 11th International Symposium on Comm. and Information Technologies.

Liang, C., Yu, F. R. (2014) “Wireless Network Virtualization: A Survey, Some Research Issues and Challenges”, IEEE Communications Survey and Tutorials.

Min, A.W. et al. (2011), "Opportunistic spectrum access for mobile cognitive radios", Proceedings of IEEE INFOCOM, pp. 2993-3001.

Nakauchi , K. et al. (2011) “AMPHIBIA: A Cognitive Virtualization Platform for End-to-End Slicing”, IEEE International Conference on Communications (ICC).

Shannon, C.E.,(2001) “A Mathematical Theory of Communication”, Association for Computing Machinery, Vol. 5, No. 1, pp. 3-55.

Wang, C. et al. (2009) “Network Selection in Cognitive Radio Systems”, IEEE Global Telecommunications Conference, (GLOBECOM).

Wang, X., Krishnamurthy, P., Tipper, D. (2013) “Wireless Network Virtualization”, International Conf. on Computing, Networking and Comm.

Xin, C., Song, M.(2012) “Dynamic Spectrum Access as a Service”, Proceedings IEEE INFOCOM.

Yang , M. et al. (2013) “Opportunistic Spectrum Sharing Based Resource Allocation for Wireless Virtualization”, Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS).

Zaki, Y. et al.(2010) “LTE wireless virtualization and spectrum management”, Third Joint IFIP Wireless and Mobile Networking Conference.

Zhang, S. et al. (2012) “An Opportunistic Resource Sharing and Topology-Aware mapping framework for virtual networks”, Proceedings of IEEE INFOCOM.

Zhao, L. et al. (2011) “LTE Virtualization: from Theoretical Gain to Practical Solution”, 23rd International Teletraffic Congress (ITC).

Trilha Principal do SBRC 2016
Sessão Técnica 22
Gerenciamento e Roteamento em SDN

Green Service Levels in Software Defined Networks

Bruno B. Rodrigues[1], Marco A. T. Rojas[1]
Viviane T. Nascimento[1], Tereza C. M. B. Carvalho[1] and Catalin Meirosu[2]

[1]Escola Politécnica – University of São Paulo (EP-USP)
São Paulo, Brazil

[2]Ericsson Research
Stockholm, Sweden

{brodrigues,matrojas,vianetn,carvalho}@larc.usp.br
catalin.meirosu@ericsson.com

Abstract. *The growing energy consumption has become a major concern for network service providers not only to reduce operational expenses, but also to minimize environmental problems related to the high energy expenditure. In this regard, assessing the energy consumed has become an important task to maximize energy efficiency, and thus, offer different green service levels for customers that desire to save energy. An SDN controller is presented that manages distinct green service levels based on power models to account for the consumed energy and energy saving capabilities. Specifically, we present three SLAs with distinct requirements in terms of energy consumption and models to account for power consumption for each network scenario. Our approach provides a fine-grained accounting of user power consumption in distinct network usage scenarios in order to increase the accuracy of GreenSLA management in next generation networks. The proposal is validated by emulating two use cases inspired on the Facebook topology.*

1. Introduction

Driven by the increasing number of users with broadband and mobile access as well as the availability of new services and experiences, the power demand of the infrastructure has become a major concern for Network Service Providers (NSPs). Services such as video streaming are driving the way network infrastructures are being designed, constantly imposing higher constraints on performance and on availability requirements. Fulfilling such requirements incurs not only in high CApital and OPerational Expenses (CAPEX and OPEX) but also leads to significant Green House Gases (GHG)[1] emissions. Ericsson presented an overview in a mobility report [Ericsson 2014], in which the number of ICT (Information and Communication Technologies) devices are estimated to increase from 6 billion in 2013 to 12.5 billion devices in 2020, being one of the main reasons of the increased carbon footprint by ICT. Figure 1 summarizes the scenario for ICT fixed and mobile networks.

Costs related to networking are among the most significant for NSPs to absorb [Ericsson 2014]. For fixed ICT networks the share of global GHG emission is estimated

[1]GHG: gasses capable of absorbing infrared radiation, trapping heat in the atmosphere and making the Earth warmer.

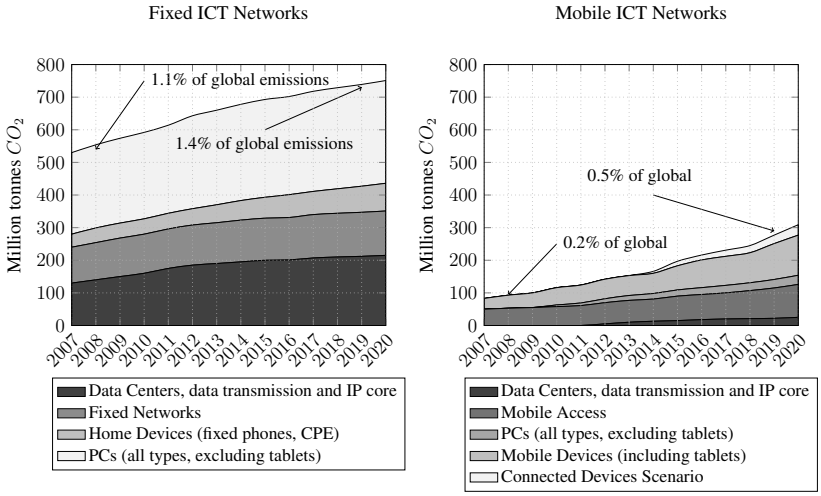


Figure 1. Global Emissions from ICT [Ericsson 2014].

to be 1.4% in 2020, and 0.5% for mobile networks, nearing 2% of the global carbon footprint. In this regard, telecom operators started to adopt environmentally responsible policies and deploy equipment that supports energy management features. A significant example comes from the Verizon 2013 Sustainability Report [Verizon 2013], is indicated that achieving energy efficiency has become critical to the ability to offer new capabilities and services, stating that 92% of the carbon emissions were due to power consumed to run their networks.

Besides, as organizations and governments around the world promote incentive programs to encourage the reduction of carbon emissions, an increasing number of customers are expected to require services tailored to specific energy conditions. However, instead of a one-size-fits-all class of service, in which energy savings and carbon credits are equally divided among customers, NSPs should offer off-the-shelf services in distinct categories of green services. In this regard, proposals were made to establish green services for data centers, ranging from green metrics (e.g., Power Usage Effectiveness - PUE and Data Center Efficiency - DCE) and services [Belady 2008, von Laszewski and Wang 2010, Le et al. 2010], to optimization frameworks [Hasan et al. 2014, Amokrane et al. 2015]. However, they do not address particularities of accounting and managing green services in a network infrastructure. As SDN facilitate the network management providing a software layer to control data plane nodes, this approach is used in this paper to implement the Green Plans.

We propose the GPController (Green Plans Controller), an SDN controller to manage the energy consumed at a user-level for network and compute resources. As main contributions, we outline the architecture and components that manage energy consumption in a fine-grained level, and power models to calculate energy consumption and savings for distinct network conditions. The proposal is evaluated based on the emulation of network and compute resources in Mininet, considering two use cases implemented on a topology inspired by the Facebook data center [Alexey Andreyev 2014]. The remainder of this work is organized as follows. Section 2 presents the related works highlighting its contribution. Section 3 presents the definition of green plans. Section 4 presents the architecture and implementation details. Section 5 presents the experimental evaluation

of Green Plans. Section 6 presents the final remarks and future works.

2. Related Work

A service is termed 'green' when deployed in computing systems designed to optimize energy efficiency and minimal environmental impact. Such services usually operate by relaxing traditional performance-based parameters for creating opportunities to save energy. In this context, Green Service Level Agreements (SLAs) are offered alongside regular SLAs, using eco-efficiency as a differentiating factor. Recently, several works have studied GreenSLAs ranging from the proposal of green metrics to optimization frameworks considering the availability of renewable energy. However, they do not address particularities on how to provide GreenSLAs for network infrastructures.

[von Laszewski and Wang 2010] introduced a framework including information about energy characteristics, focusing on energy metrics for green services (i.e., DCiE, PUE). Based on this, [Le et al. 2010] introduced a general, optimization-based framework and several request distribution policies enabling multi-data-center services to manage their brown energy[2] consumption and leveraged green energy, while respecting their SLAs. [Klingert et al. 2011] introduced the notion of GreenSLAs, identifying known hardware and software techniques to reduce energy consumption and to integrate green energy. In a case study, the authors compared three types of SLA: (i) a performance-based one that does not address energy consumption prioritizing performance and time; (ii) a relaxed SLA that requires key indicators to be within relaxed boundaries, and (iii) an energy-aware SLA that uses tight energy ranges for each job.

[Bunse et al. 2012] conducted a case study considering the SLA types proposed by Klingert et al. and providing an experimental evaluation of energy management in the context of web services and mobile clients. [Haque et al. 2013], proposed GreenSLAs in which a customer specifies a percentage of green energy to be used in the execution of data center workloads (e.g., x% of the job should run on green energy). Instead of a per job/application approach, [Hasan et al. 2014] proposed a specific time interval considering availability and price combination to buy green energy from the market.

[Hasan et al. 2015] continues to investigate the negotiation between green energy sources and cloud service providers, yet providing means to ensure that a data center can be proportionally green for the whole day, by exploiting available green energy sources and markets. [Amokrane et al. 2015] proposed a resource management framework, allowing providers to create a virtual infrastructure to provide resources (i.e., a set of virtual machines and virtual links with guaranteed bandwidth) across a geo-distributed infrastructure.

Most related GreenSLAs approaches consider the proposal of optimization frameworks within a triangle of energy, QoS, and cost to establish green services for data center resources. However, such approaches neglect particularities of networking services, and more specifically SDN networks, to propose differentiated green services. Inspired by the aforementioned approaches, we focus on enabling green services for SDN networks including a partial study on compute resources.

[2]Energy produced by polluting sources.

3. Green Plans

As governments and organizations promote incentives to reduce the carbon footprint, an increasing number of customers are expected to demand green products and services. For NSPs, this is not different. On the customer side, it is desirable to contract services in which the performance level can be adjusted according to daily requirements, defining time frames in which the performance level can be increased or decreased (e.g., increase the performance level during office hours, otherwise decrease outside office hours). On the NSP side, besides providing hardware and software capabilities to increase the energy efficiency, a system is required to manage the energy consumption for green customers. According to [Klingert et al. 2011], we defined three classes of service in Table 1.

Plan	**Network Resources**					**Compute Resources**	
	Bandwidth (Mbps)	**Delay (Ms)**	**Jitter (Ms)**	**Packet Loss (%)**	**Max. Energy (Watts)**	**vCPU (%)**	**Max. Energy (Watts)**
BP	B_1	D_1	J_1	PL_1	NE_1	C_1	CE_1
GP1	B_2	D_2	J_2	PL_2	NE_2	C_2	CE_2
GP2	B_3	D_3	J_3	PL_3	NE_3	C_3	CE_3

Table 1. Green Plans Parameters.

BP (Brown Plan) is a full-performance plan, in which customers have the best possible configuration in terms of performance. GP1 (Green Plan 1) represents a mid-term between performance and energy savings, and GP2 (Green Plan 2), an energy saving plan for customers desiring to save energy. Considering devices in which the energy consumed is proportional to the workload to be processed, we distinguished green plans based on provisioned $Bandwidth$ and $vCPU$. $Bandwidth$ considers a hierarchy $B_1 > B_2 > B_3$, meaning that BP customers have more bandwidth available than GP2 and so on. Since $Delay$, $Jitter$ and $Packet\ Loss$ relies on applications, they are selected in accordance with requirements imposed by applications. The $vCPU$ is the maximum compute workload for each host. Similarly to $Bandwidth$, the values consider $C_1 > C_2 > C_3$ to measure the energy consumed by servers.

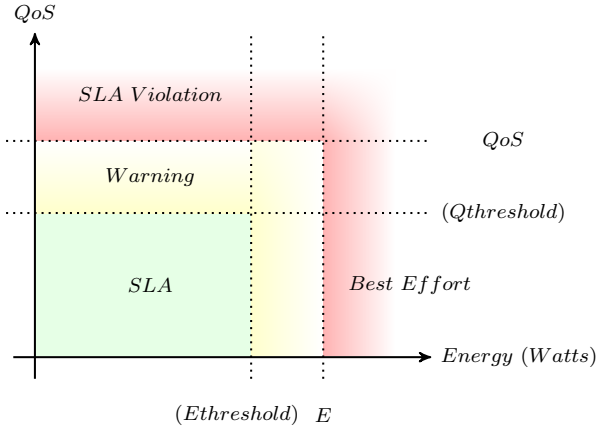


Figure 2. Thresholds for QoS and energy consumption.

Besides QoS parameters, we specified the amount of energy to be consumed by both networks (NE_i) and compute resources (CE_i). Therefore, customers desiring to save energy may set thresholds for the energy consumed. Thus, if the energy consumed reaches NE_i, or CE_i the customer enters a best effort mode until a new plan is contracted. Figure 2 illustrates the thresholds for both QoS and energy parameters. Based on QoS and energy parameters, alarms are set to define thresholds for both QoS ($QoS-k$) and energy ($E-k$). In the warning zone, policies can be employed to either renew the energy consumed for compute or network resources, or to adjust the QoS levels. Power capping is used when a customer reaches its NE_i or CE_i thresholds entering a best effort zone, in which its service levels are reduced. This occurs either by using traffic shaping capabilities or limiting the vCPU capacity.

IF eConsumed $> eThreshold$ **AND** eConsumed $\leq$ E **THEN**: Send alarm to customer

4. GPController Architecture

This section provides details of green services deployment in SDNs, presenting architectural and implementation solutions employed for GPController. Based on the ONF (Open Networking Foundation) SDN architecture, the GPController comprises network and compute resources structured in four abstraction planes: i) data, ii) control, iii) application, and iv) management. Figure 3 presents the architecture.

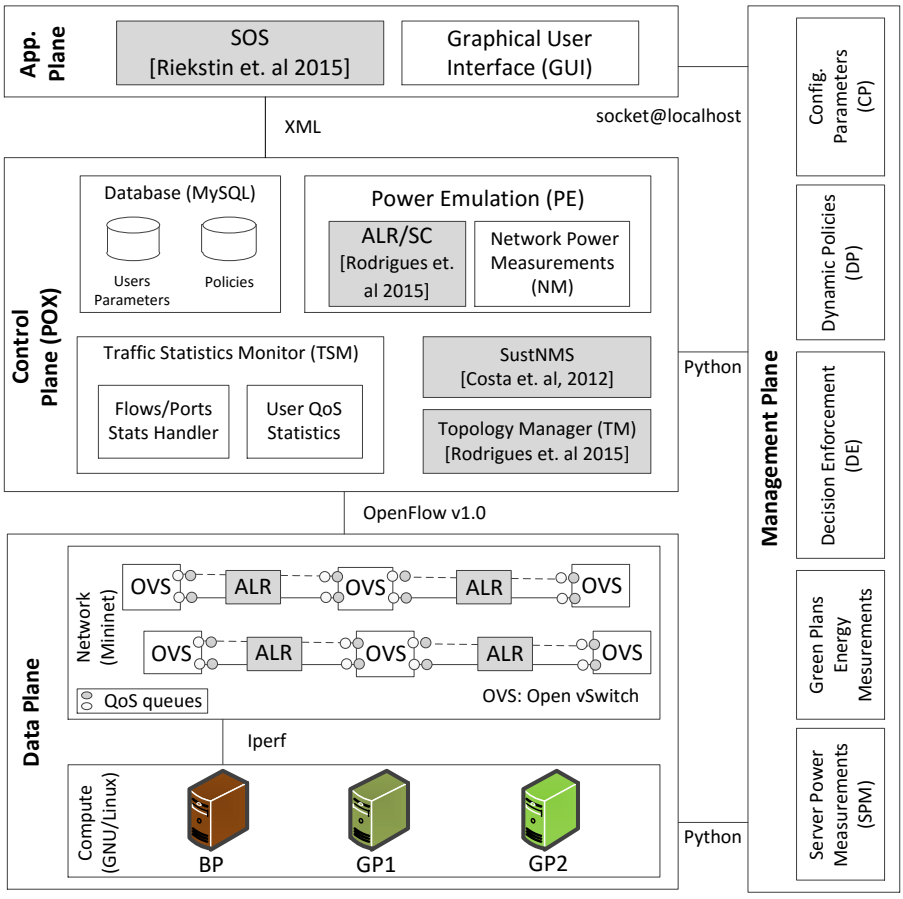


Figure 3. GPController Architecture

- **Application plane** comprising the SOS orchestration method and Graphical User Interface (GUI) elements, such as a topology viewer, statistics charts, to obtain user parameters;
- **Management plane**: interconnect the planes receiving information and taking decisions on which energy efficiency capabilities to employ and users requirements regarding energy consumption;
- **Control plane**: presents modules to obtain and to prepare data for the management layer, such as the Topology Manager (TM) that contains the network graph and controls flow tables; Traffic Statistics Monitor (TSM) to query nodes ports/flows statistics; Power Emulation (PE) to measure energy consumption; Database Manager to hold network and users logs; and the SustNMS capability to perform green traffic engineering; and
- **Data plane**: comprising Open vSwitch nodes configured with QoS queues to perform a traffic shaping capability, and servers based on virtual cores of the machine host. To generate network traffic we used the Iperf[3], and Cpuloadgen[4] to generate compute workload for each virtual core.

The GPController operates in two stages, configuration, and management. Configuration is related to parsing and deploying inputs from SOS and users. SOS performs a training stage before the GPController starts, to decide on the best combination of energy efficiency capabilities to be activated given a network utilization. As a result, it produces decision trees and policies describing environment and time conditions to change or to update decision trees. SOS parameters are parsed via XML (eXtensible Markup Language) files for configuring the Decision Enforcement (DE) and the Dynamic Policies (DP) modules. Users are able to define QoS and energy parameters using a GUI. Then, this information is sent via socket for configuring the databases with users policies and requirements.

After the deployment of SOS and users parameters, the management stage begins. It is composed of three steps: a) query data plane information regarding port/flow statistics, b) measurement of energy and QoS parameters, and c) enforcement of decisions. The Traffic Statistics Monitor (TSM) is responsible for querying node statistics regarding flow and ports usage. Compute resources are monitored by the Server Power Measurements (SPM) module at the management plane. It coordinates a workload generator (Cpuloadgen) capable of generating workload on each virtual core and comprises models to calculate the energy consumed by servers based on generated workload.

The PE module comprises models for each possible energy state in which nodes can be. Upon receiving bandwidth information for each active user, it calculates the energy consumed and saved by verifying current node's states (e.g., active, sleeping, or saving energy) for each node in the user path and enforcing the respective power model. QoS values are obtained by injecting probe packets in flows configured for each user at a regular time interval (detailed in subsection 4.1). Next, both QoS and energy information regarding each user, as well as an overall network are stored. As the final step in the management cycle, the DE (Decision Enforcement) query statistics to assess whether it is necessary to adjust the performance of the network or to compute resources. Two distinct

[3]https://iperf.fr/

[4]https://github.com/ptitiano/cpuloadgen

decisions are performed. The first aims to ensure users requirements regarding energy and QoS parameters. The second is based on a holistic network view, related to enforcing decisions given by SOS decision trees (i.e., given an overall network utilization decides whether to enforce energy efficiency capabilities).

4.1. Obtaining Data Plane Information

To match user packets and to account for network statistics, we used a MAC address flow instantiation. Based on prior knowledge of user's routes, two distinct rules were used to avoid the potential flood of rules in flow tables. One for edge nodes, specifying source and destination MAC address, and the other for interconnection nodes specifying destination MAC address. Furthermore, to proportionally calculate user's statistics accounting for the number of nodes that are shared among users is required. Thus, we account nodes in the path of active users, maintaining a dictionary of counters for each node. Once a user is inactive, counters of nodes in his/her path are decremented in the dictionary. Next, we provide details on how to obtain delay, jitter and packet loss in a per-user basis:

- **Delay**: given prior knowledge of user's path, whenever a workload is sent by an user probe packets are forwarded to the destination node from the source node with a timestamp as payload. Then, the switch-to-controller delay is estimated by determining its RTT (Round-Trip Time) injecting packets that are immediately returned to the controller, dividing the RTT by two to account for the bi-directionality of the given answer (Equation 1):

$$Delay = (t_{arrival} - t_{sent} - \frac{1}{2}(RTT_{src} + RTT_{dst})) \tag{1}$$

- **Jitter**: calculated as the average absolute value of the difference of consecutive delay samples. Given at least two consecutive delay samples (Equation 2), it is calculated as the average of absolute values of difference in two consecutive delay samples in a period of measuring time (Equation 3):

$$DelaySamples = [t_i, t_{i+1}, ..., t_k] \tag{2}$$

$$Jitter = abs\left(\sum_{t=1}^{T_k}(t_i - t_{i+1}) + (t_{i+1} - t_k)\right) \tag{3}$$

- **Packet loss**: estimated by polling flows statistics from source and destination nodes of each path. Uses delay probe-packet and control flags (src_flag and dst_flag) to detect when to subtract statistics from destination and source. When a probe packet is sent, src_flag is marked as true, and upon the packet's arrival, dst_flag also is marked as true. When both are true, the loss is calculated by subtracting the increase of the source switch packet counter from the increase of the packet counter of the destination switch. Then, both flags are set as false and another measuring round can be started.

To emulate compute workload on a per-user basis, a tool to generate workload on each core of the host machine was used (Cpuloadbench). Despite achieving a fine-grained generation of workload, a virtual core for each user has to be created, thus imposing a scalability restriction. Whenever a workload is sent, the SPM generates a random workload

between a fixed range (e.g., Brown Plan between 50-70% of CPU utilization) for each green plan. Then, a process is spawned to run the micro-benchmark. The CPU utilization of spawned processes is monitored and obtained the values are used in power models to calculate the energy consumed.

4.2. Green Plans Power Models

The component calculates the energy consumed and saved from users considering the state of nodes in the user path. Information on current workload and path are received from the TSM, and regarding servers consumption from the SPM. Then, the component checks the states of nodes in the user path, applying a specific power model. To obtain the energy consumed for network resources we consider the power models presented in [Rodrigues et al. 2015]. Compute resources consider a model in Equation 4 describing the energy consumed by a server including CPU, memory and disk resources [Beloglazov et al. 2012]:

$$P(u) = \underbrace{k}_{70\%} * \underbrace{P_{max}}_{250\ \text{W}} + (1-k) * P_{max} * u \tag{4}$$

The $Pmax$ is the maximum power consumed when the server is fully utilized; k is the fraction of the power consumed by the idle server (i.e. 70%), and u is the CPU utilization. For our experiments, P_{max} is set to 250W according to [Beloglazov et al. 2012].

For network resources, the energy consumed is calculated by verifying states (i.e., active, sleeping, or applying an energy efficiency capability) of nodes in the user path and using the referred power model. However, such models do not calculate the energy consumption in a user granularity. Given that the energy consumed by the node chassis (comprising internal components such as CPU, RAM, fans) is not load proportional, we defined a model (Equation 5) in which the energy consumed by the node chassis ($P_{chassis}$) is distributed in accordance with the user workload and maximum link capacity. Thus, when distinct users are sharing a node in a certain measurement point in time, the energy consumed by the chassis is proportionally distributed among the users.

$$PP'_{on} = \underbrace{\left(\frac{P_{chassis} * W_{user}}{LinkCapacity}\right)}_{\text{Pchassis proportional energy distribution}} + \sum_{i=1}^{nPorts} (P_{port} * W_{user}) \tag{5}$$

$$PP'_{sleep} = \left(\frac{120}{NumGreenUsers}\right) \tag{6}$$

$$PP'_{ALR} = PP'_{on} - 15\% \tag{7}$$

$$PP'_{SC} = PP'_{sleep} * \left(\frac{tOn}{DutyCycle} - tOn\right) + PP'_{on} * tOn \tag{8}$$

Considering that several users can share the same nodes, their consumption can be obtained by splitting the fixed (representing the internal components such as CPU, memory, fans) consumption part among them. In this regard, Equation 5 is applied when a node is active, Equation 7 when ALR or SC is being applied. The energy consumed by users is measured as follows:

$$A_c\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{on}\ (W_{user}) \quad (9)$$

A_c: consumption from nodes powered on

$$B_c\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{(ALR\ or\ SC)}\ (W_{user}) \quad (10)$$

B_c: consumption from nodes enforcing ALR or SC

$$C_c\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{sleep} \quad (11)$$

C_c: consumption from nodes sleeping

$$D_c\ (W) = A + B + C \quad (12)$$

In A_c the energy consumed from active nodes is obtained. B_c calculates the number of Watts consumed from nodes applying either ALR or SC. C_c returns the Watts consumed for sleeping nodes. As SustNMS requires concentrating the traffic on a certain path while unused nodes are put to sleep, energy savings from affected users are obtained from nodes in sleep mode. In the last step, D_c performs the sum of the user's consumption. Savings per user is obtained by comparing their consumption with the maximum workload allowed in the network (W_{max}). Energy savings per user is measured as follows:

$$A_s\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{on}(W_{max}) - A_c \quad (13)$$

A_s: savings nodes powered on

$$B_s\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{on}(W_{max}) - B_c \quad (14)$$

B_s: savings nodes enforcing ALR or SC

$$C_s\ (W) = \sum_{i=1}^{\mathbf{N}_{switches}} PP'_{on}(W_{max}) - C_c \quad (15)$$

C_s: Savings nodes sleeping

$$D_s\ (W) = A + B + C \quad (16)$$

D_s: Sum of savings

$$S\ (\%) = (D_s * 100)/D_s \quad (17)$$

The difference between the energy saving models and the energy consumption models is the consumption with the user workload subtracted from the consumption of the maximum workload used as a reference. To illustrate the operation of the module, Algorithm 1 presents the consumption measurements.

5. Experimental Evaluation

The goal of this evaluation is to assess the architecture and power models, as well as mechanisms to ensure a constraint regarding the amount of energy to be consumed. The GPController was deployed in a VM configured with 4 virtual cores and 4 GB RAM. The host machine is an Intel Core i5-3570 @ 3.40GHz with 8 GB RAM. The SDN network was emulated in Mininet and the GPController is based on the POX controller. Network traffic is generated through the Iperf, which is already available in Mininet. To create a separate workload for each virtual core, compute workload was emulated using Cpuloadbench. Thus, the emulation is restricted to 4 users, which is the number of virtual cores.

Algorithm 1: Algorithm to calculate the energy consumed and saved by users.

Input: active_hosts ⟵ list of active users
Input: sharedNodes ⟵ dictionary nodes shared by users
Output: Energy consumed (W) and savings (%) per user

1 **begin**
  /* Loop active users */
2 **for** $each\ user \in active\_hosts$**:**
    /* Loop nodes in the user path */
3   **for** $each\ node \in user.path$**:**
      /* Calculate the Energy Consumed */
4      $A_c \longleftarrow user.workload,\ node,\ sharedNodes$
5      $B_c \longleftarrow user.workload,\ node,\ sharedNodes$
6      $C_c \longleftarrow user.workload,\ node,\ sharedNodes$
      /* Calculate the Energy Saved */
7      $A_s \longleftarrow user.workload,\ node,\ sharedNodes$
8      $B_s \longleftarrow user.workload,\ node,\ sharedNodes$
9      $C_s \longleftarrow user.workload,\ node,\ sharedNodes$
    /* Sum of the energy consumed */
10    $D_c \longleftarrow A_c + B_c + C_c$
    /* Sum of the energy saved */
11    $D_s \longleftarrow A_s + B_s + C_s$
    /* Percentage of the energy saved */
12    $S \longleftarrow D_s * 100/D_s$
    /* Store the result */
13    $energyCS[user] \longleftarrow [D_c, S]$
14 **Return** $energyCS$

5.1. Evaluation Settings

The topology is inspired by the Facebook data center fabric [Alexey Andreyev 2014]. To emulate ALR (which adjusts the link rate according to current load) we interconnected each pair of nodes using parallel links, which are configured with different rate limits. Standard links are configured to handle a maximum traffic of 30 Mbps, and ALR links 10 Mbps.

To send data across the network, four hosts were placed at node 21 and two sinks in nodes 1 and 27. Considering the Facebook scenario presented in [Alexey Andreyev 2014], two cases were considered for setting up flows. The first considers traffic going out of the data center (user to machine), and the second internal traffic (machine to machine). As more people connect to the Internet and new products and services are created, the user to machine traffic is large and ever increasing. However, machine to machine traffic is several orders of magnitude larger than that going out to the Internet (e.g., a simple scroll in a Facebook timeline require internal data that is spread internally within the data center servers). Furthermore, settings used in the evaluation are summarized in Table 2.

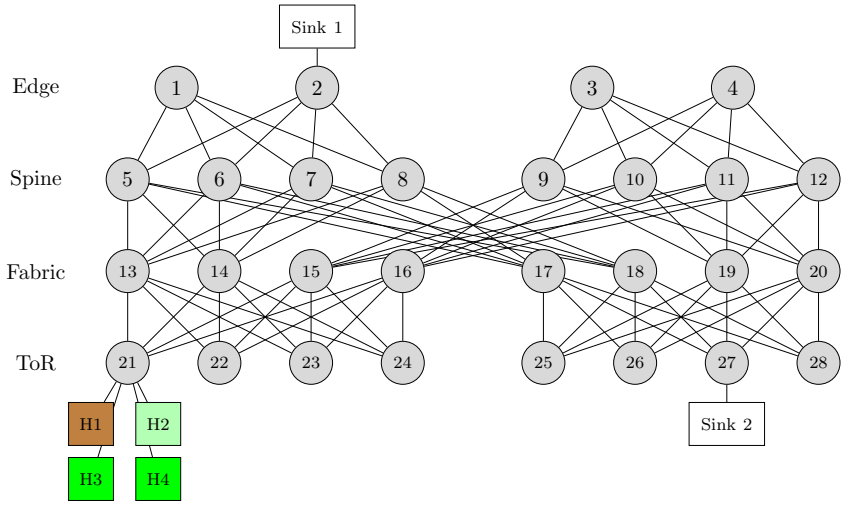


Figure 4. Topology of the evaluation system.

User	Case 1 Path	Case 2 Path	Bandwidth (Mbps)	vCPU (%)	Max. Energy C_E / N_E (Watts)
BP	[21-13-5-2]	[21-13-5-2]	15	70 - 100	1.5 / 14
GP1	[21-13-5-2]	[21-15-10-19-27]	9	30 - 40	1 / 10
GP2a	[21-13-5-2]	[21-15-10-19-27]	3	20 - 30	0.8 / 4
GP2b	[21-13-5-2]	[21-15-10-19-27]	3	20 - 30	0.8 / 4

Table 2. Evaluation settings for Cases 1 and 2.

Since the maximum link capacity was set to 30 Mbps, we divided the maximum reachable bandwidth for each user which was divided among users to provide a 100% link utilization in case four users share the same path. Thus, we considered 15 Mbps for a BP user, 9 Mbps for GP1, and 3 Mbps for GP2 (placing two GP users). The distribution of computational load in each vCPU allocated for users follows the same logic. For a BP user, the workload generated simulates an application that requires intensive processing, between 70 and 100% of the vCPU. However, the workload for GP1 and GP2 users is lower to avoid a computational overhead in the experiment. Thus, workload ranges between 30-40% for GP1 users, and 20-30% for GP2 users.

5.2. Evaluation Results

Figure 5 presents energy consumption and savings results in function of time for both use cases and employed energy efficiency capabilities. In Case 1, in which users share the same path [21-13-5-2], energy savings are distributed proportionally among green users considering nodes that were put in sleep mode. For instance, since the traffic engineering capability (SustNMS) performs a load balance using an alternate path to the predefined route [21-14-6-2] towards Sink 1, we forced users in the same route to evaluate QoS results with full link utilization. In this case, the BP user does not save energy once it uses the maximum bandwidth, however, savings from nodes in sleep mode are distributed among green users (GP1 and two GP2).

For Case 2, the BP user was configured with distinct routes from GP1 and GP2 users; thus individual decisions on energy efficiency capabilities were made due to the

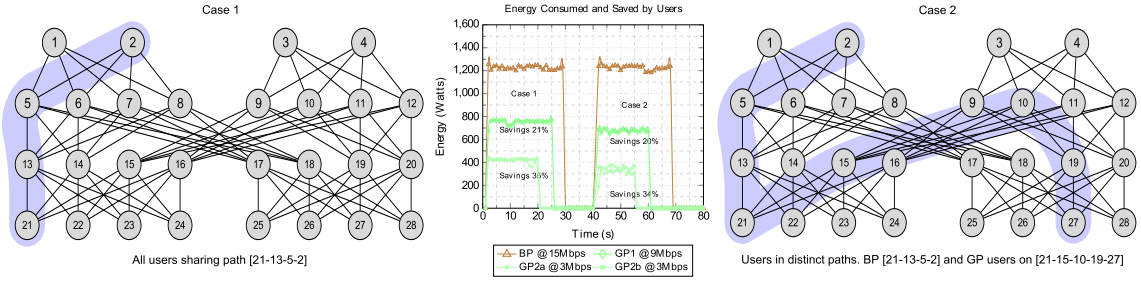


Figure 5. Per user energy consumption and savings and energy efficiency capabilities employed in both cases.

low utilization of nodes [15-10-19-27]. Despite GP2 users consuming a smaller amount of energy, they presented lower energy savings than in Case 1 because more nodes were active. Besides employing energy efficiency capabilities on such nodes, green traffic engineering is still more effective to save energy, aggregating traffic and putting unused nodes to sleep.

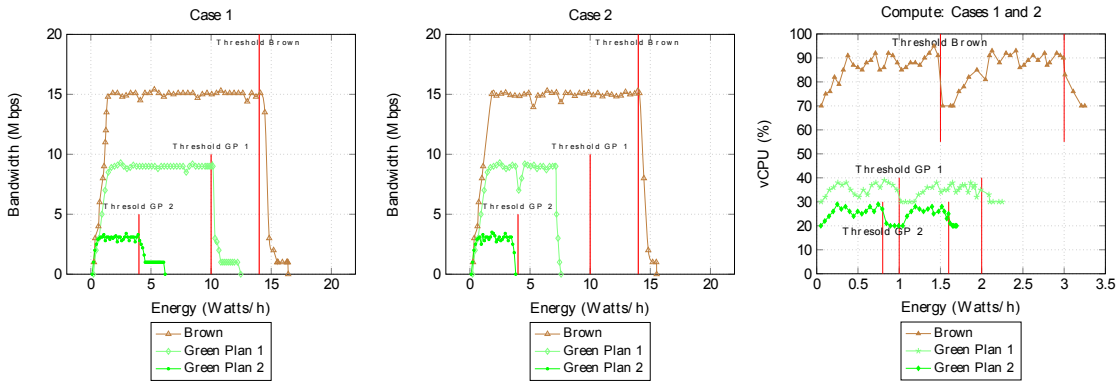


Figure 6. Max energy consumption thresholds

To evaluate energy threshold policies, random C_E and N_E values were defined. Figure 6 presents the energy consumed (W/h) per users in function of their workload, and aggregated to compute resources in a single run. When N_E is reached, user's traffic is forwarded onto a best effort QoS queue. Since the BP and GP1 users had the same bandwidth and routes for both cases, results on network energy thresholds were similar. For GP2 users, due the enforcement of SC+ALR capabilities on nodes [15-10-19-27] at the Case 2, the energy consumed did not reach the energy threshold. Conversely to network resources, we considered a single continue run for compute. Since the consumption relies on vCPU workload and the overall performance of the host machine, C_E thresholds were adjusted to force the GPController to restrict the vCPU usage. When C_E is reached, the workload generator component limits the workload on Cpuloadbench. Since we did not consider any energy saving feature for compute resources, thresholds for both cases were reached.

The quality of service effects of link utilization are presented in Figure 7. As in Case 1 users were configured to forward packets through the same path - ignoring the SustNMS load balance, their QoS statistics was bad contrasting with the Case 2. Thus, with 100% of link occupancy the segment between nodes 21 and 13 was saturated and

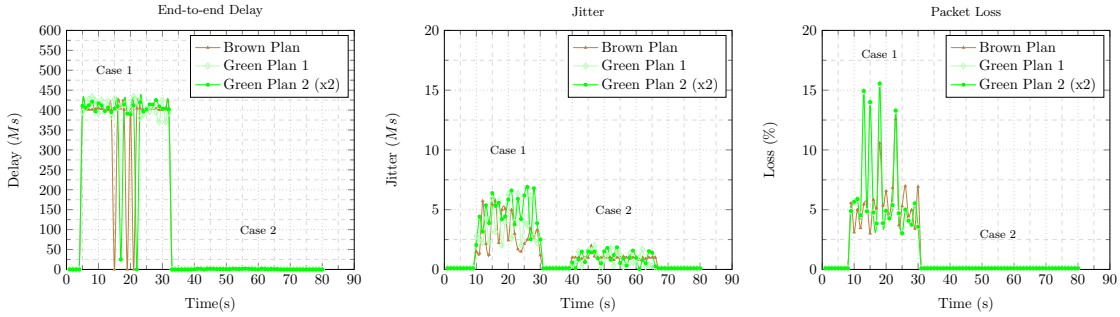


Figure 7. QoS results cases 1 and 2.

packets started to be queued and dropped. Since in Case 2 users were configured with distinct routes, links were not oversubscribed and QoS statistics was satisfactory for all users.

6. Final Considerations and Future Works

An SDN controller based on network energy efficiency capabilities was presented being a first step towards establishing green service levels for SDN networks. Thus, inspired on GreenSLAs works we presented an SDN architecture and power models to manage the usage of network and compute resources by users. Despite providing a closer view on a real deployment details, a validation by emulation imposes restrictions on scalability. However, it can provide a closer view of functioning details of the experiment, usually neglected by simulation. As future work, we consider combining emulation and simulation to scale the number of users at the different plans, as well as adjustments to the decision enforcement point to configure the network in case of QoS violations. For the Green Plans, we consider the deployment of policies in which users may define time frames to increase or to decrease performance levels by changing between green plans. Thus, allowing to adjust resources provisioning according to periods in which they are most required.

Acknowledgments

This work was supported by the Innovation Center, Ericsson Telecomunicações S.A., Brazil.

References

Alexey Andreyev (2014). Introducing data center fabric, the next-generation facebook data center network. Technical report, Facebook.

Amokrane, A., Langar, R., Faten Zhani, M., Boutaba, R., and Pujolle, G. (2015). Greenslater: On satisfying green slas in distributed clouds. *Network and Service Management, IEEE Transactions on*, PP(99):1–1.

Belady, C. (2008). Data center power efficiency metrics: Pue and dcie.

Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768.

Bunse, C., Klingert, S., and Schulze, T. (2012). Greenslas: Supporting energy-efficiency through contracts. *Energy Efficient Data Centers*, page 54.

Ericsson (2014). Ericsson Energy and Carbon Report - Including Results from the First-ever National Assessment of the Environmental Impact of ICT.

Haque, M. E., Le, K., Goiri, Í., Bianchini, R., and Nguyen, T. D. (2013). Providing green slas in high performance computing clouds. In *Green Computing Conference (IGCC), 2013 International*, pages 1–11. IEEE.

Hasan, M., Kouki, Y., Ledoux, T., and Pazat, J. (2014). Cloud energy broker: Towards sla-driven green energy planning for iaas providers. In *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS), 2014 IEEE Intl Conf on*, pages 248–255.

Hasan, S., Kouki, Y., Ledoux, T., and Pazat, J. (2015). Exploiting renewable sources : when green sla becomes a possible reality in cloud computing. *Cloud Computing, IEEE Transactions on*, PP(99):1–1.

Klingert, S., Schulze, T., and Bunse, C. (2011). Greenslas for the energy-efficient management of data centres. In *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*, pages 21–30. ACM.

Le, K., Bilgir, O., Bianchini, R., Martonosi, M., and Nguyen, T. D. (2010). Managing the cost, energy consumption, and carbon footprint of internet services. In *ACM SIGMETRICS Performance Evaluation Review*, volume 38, pages 357–358. ACM.

Rodrigues, B., Riekstin, A., Januario, G., Nascimento, V., Carvalho, T., and Meirosu, C. (2015). GreenSDN: Bringing Energy Efficiency to an SDN Emulation Environment. In *Integrated Network and Service Management (IM), 2015 14th IFIP/IEEE Symposium on*.

Verizon (2013). 2013 Corporate Responsibility Supplement. Technical report, Verizon.

von Laszewski, G. and Wang, L. (2010). Greenit service level agreements. In *Grids and Service-Oriented Architectures for Service Level Agreements*, pages 77–88. Springer.

Gerenciamento Flexível de Infraestrutura de Acesso Público à Internet com NFV

Alexandre Heideker, Carlos Kamienski

Universidade Federal do ABC (UFABC)
{alexandre.heideker, cak}@ufabc.edu.br

Abstract. *The concepts of Smart Cities, Internet of Things (IoT), smartphones and other pervasive computing technologies are reality, creating computing and telecommunications infrastructure demands, requiring flexibility, quality of service and reducing financial and environmental cost. Technologies like Network Functions Virtualization (NFV), Cloud Computing and Software-Defined Network (SDN) appear as a promise to address these demands. This work uses NFV to solve the problem of dynamic and elastic supply of public Internet access in big cities. The results show significant levels of performance and flexibility, and performance metric comparison between KVM, XEN and LXC containers in NFV.*

Resumo. *Os conceitos de Cidades Inteligentes, Internet das Coisas (IoT), smartphones e outras tecnologias de computação pervasiva, já são realidade, gerando demandas computacionais e de infraestrutura de telecomunicações, exigindo flexibilidade, qualidade de serviço e redução de custo financeiro e ambiental. Tecnologias como Virtualização de Funções de Rede (NFV), Computação em Nuvem e Redes Definidas por Software (SDN) figuram como promessa para solucionar essas demandas. Este trabalho utiliza NFV para solucionar o problema de fornecimento dinâmico e elástico de acesso público à internet nas grandes cidades. Os resultados obtidos demonstram níveis significativos de desempenho e flexibilidade, além de métricas de comparação de desempenho entre KVM, XEN e containers LXC em NFV.*

1 Introdução

O processo de urbanização e a adoção de tecnologias altamente conectadas, como *smartphones*, *tablets*, Internet das Coisas (IoT), casas inteligentes, redes veiculares, entre outras, desencadearam novos desafios, tanto no que concerne ao processamento desses dados como no transporte destes por intermédio das redes de computadores. Este arcabouço de problemas e soluções deu origem à área da computação urbana, que trata o desafio das cidades inteligentes com uma abordagem computacional [Zheng et al. 2014].

Entre as abordagens utilizadas para tratar o problema computacional das cidades inteligentes, destaca-se a computação em nuvem, que permite a alocação de recursos sob demanda, reduzindo investimento e apresentando-se como solução ambientalmente responsável. Outro paradigma oriundo deste cenário é o *Fog Computing*, onde a união de nuvens públicas e privadas, computação embarcada, IoT entre outros elementos da computação urbana formam um grande sistema [Stojmenovic and Wen 2014].

O tratamento do desafio de transporte dos dados deste ambiente também exige soluções inovadoras, baratas e ambientalmente corretas – mobilidade e elasticidade são premissas básicas para soluções de telecomunicações neste ambiente. A virtualização de

redes, utilizando tecnologias como SDN *(Software-Defined Networking)*[McKeown et al. 2008] e NFV *(Network Functions Virtualization)* [ETSI et al. 2012] são grandes promessas para a solução dessas demandas.

Entre os muitos desafios apresentados por este novo ambiente de convivência do cidadão, um dos mais presentes em seu cotidiano é o acesso à internet em locais públicos, política que já vem sendo adotada por diversas cidades ao redor do mundo utilizando a tecnologia Wi-Fi. A partir de 2014, a Prefeitura de São Paulo deu início ao projeto WiFi Livre SP[1], com o objetivo de fornecer acesso público à internet em 120 praças da cidade [Ratusznei et al. 2014], utilizando tecnologias e topologias tradicionais de rede. Além de questões socioculturais apresentadas por esta iniciativa, a qualidade do serviço oferecido também representa um grande desafio neste projeto.

Este trabalho propõe uma solução para gerenciamento de infraestrutura de acesso público à Internet através de WiFi que usa funções de rede virtualizadas (VNF – *Virtualized Network Function*) como alternativa aos sistemas tradicionais baseados em plataformas específicas de software e hardware, abordando três das nove tendências tecnológicas para 2016 apontadas pela IEEE Computer Society[2]. Particularmente, é apresentada uma solução para o gerenciamento de NAT *(Network Address Translator)* em um ambiente com praças digitais onde milhares de usuários usam a Internet todos os dias. As funções de rede virtualizadas são criadas e destruídas, de acordo com a demanda, ou seja, é realizado um gerenciamento dinâmico de elasticidade para NFV, semelhante ao que tradicionalmente ocorre em ambientes de nuvem computacional.

Os resultados obtidos corroboram as vantagens preconizadas pelo ETSI *(European Telecommunications Standards Institute)*, no incentivo à adoção da tecnologia NFV, demonstrando que a divisão das tarefas realizadas por um *appliance* em múltiplas instâncias virtualizadas produzem resultados comparáveis às soluções tradicionais dentro dos intervalos de confiança obtidos nos experimentos. Além de apresentar esta avaliação, a mesma é realizada utilizando as técnicas de virtualização KVM, XEN e containers LXC, demonstrando sua influência no desempenho geral da solução e identificando os prós e contras na escolha dessas tecnologias.

A seção 2 apresenta os trabalhos correlatos à tecnologia NFV e Cidades Inteligentes. Na seção 3, tem-se os conceitos de NFV e as técnicas de virtualização utilizadas. A seção 4 traz a metodologia utilizada e os experimentos preliminares para definição de parâmetros e métricas. Na seção 5 são apresentados os resultados obtidos pelos experimentos e, na seção 6, é realizada a discussão sobre os resultados obtidos e lições aprendidas.

2 Trabalhos Relacionados

O fornecimento de internet em áreas públicas é explorado em [Heideker and Kamienski 2015], no qual o fenômeno *flash crowd* é explorado utilizando NFV para prover recursos de infraestrutura em demandas sazonais, utilizando o sistema operacional de nuvem OpenStack[3] para fornecer as funções de rede virtualizadas. Grande parte dos trabalhos relacionados à Computação Urbana tratam da informação contida neste sistema ou da infraestrutura de acesso individual, como em [Roth et al. 2010], onde

[1] http://wifilivre.sp.gov.br. Acesso em: 11/01/2016.

[2] http://www.prnewswire.com/news-releases/ieee-computer-society-predicts-top-9-technology-trends-for-2016-300193210.html. Acesso em: 18/02/2016.

[3] http://www.openstack.org. Acesso em: 05/01/2016.

tecnologias de acesso como WiMAX e redes 4G são discutidas, bem como o desafio do posicionamento e distribuição de antenas para obter a maior cobertura por km^2 possível. Relacionando, também, à ideia das casas inteligentes e acesso à serviços digitais, a proposta de [Bronstein 2014] trata o problema do fornecimento de internet, entretenimento multimídia, telefonia e outros recursos sob o ponto de vista da tecnologia NFV, transformando essas demandas em VNFs que podem ser alocadas de acordo com a necessidade e/ou preferência da residência.

Com o objetivo de propor uma implementação de plataforma para prover NFV, em [Soares et al. 2014] os autores propõem a criação de uma plataforma específica para NFV utilizando outra tecnologia em franca ascensão: Redes Definidas por Software (SDN - *Software Defined Network*). Essa plataforma, batizada de Cloud4NFV utiliza a ideia central da tecnologia SDN de desacoplar o plano de dados do plano de controle da infraestrutura de rede [Open Networking Foundation 2012], direcionando os fluxos de dados para a VNF responsável pela sua manipulação. Com o mesmo objetivo, em [Gember-Jacobson et al. 2015], os autores propõem a criação de um repositório de funções de rede chamado OpenNF, onde os fluxos de dados externos são manipulados e direcionados utilizando também os conceitos de SDN. O enfoque do OpenNF é a preservação do estado, ou seja, redução de retrabalho, reenvio ou descarte de pacotes durante o processo de criação, manipulação ou falha de uma VNF, deixando em segundo plano o conceito de virtualização. Além desses esforços, destaca-se os projetos ZOOM[Demchenko et al. [S.d.]], OpenMANO[4] e UNIFY [Császár et al. 2013].

Outra abordagem voltada à performance da tecnologia é explorada em [Martins et al. 2014] com o ClickOS, onde o tempo de criação de máquinas virtuais, da ordem de poucos milissegundos, e a melhora no tempo de manipulação dos pacotes é tratada com uma versão minimalista do sistema operacional, além de ajustes no hipervisor para obter o máximo desempenho possível.

O trabalho de [Bari et al. 2015], por sua vez, explora a orquestração de NFV, apresentando resultados analíticos que apontam para uma redução de até 4 vezes no custo de operação e investimento (OPEX) em relação à abordagem tradicional, formalizando a solução como um problema de programação linear.

Este trabalho diferencia-se dos demais por implementar um mecanismo de elasticidade utilizando NFV, tratando o problema de fornecimento de infraestrutura dinâmica, utilizando dados de um cenário real para avaliar o desempenho da tecnologia NFV e seu relacionamento com três tecnologias distintas de virtualização, servindo desta forma como prova de conceito para fundamentar ainda mais a tecnologia NFV.

3 Conceitos Básicos

3.1 Virtualização de Redes e NFV

O modelo de referência OSI [Sousa et al. 2007], assim como o próprio modelo TCP/IP, têm como objetivo a criação de camadas de abstração da infraestrutura de rede necessária à comunicação. Essa abstração mostrou-se eficaz e promoveu o crescimento das redes de computadores. No cenário atual, é comum que a informação, entre sua origem e seu destino, atravesse um grande número de elementos de rede responsáveis por este trajeto. Além dos enlaces, estão envolvidos nesta tarefa os chamados

[4] http://www.tid.es/long-term-innovation/network-innovation/telefonica-nfv-reference-lab. Acesso em: 13/012016.

middleboxes, entre os quais destacam-se Roteadores, NAT, VPN *(Virtual Private Network)*, *Proxy*, *Firewall*, LB *(Load Balancer)*, entre outros.

Segundo Patouni [Patouni et al. 2013], a virtualização de redes consiste na abstração destes recursos, simplificando a tarefa de alocação, considerando localização física e uso, além de promover a separação entre recursos físicos e lógicos.

O trajeto percorrido pela informação nas redes atuais, como foi mencionado, envolve outros elementos além dos enlaces. Estes *middleboxes*, sob o ponto de vista da virtualização de redes, são denominados Funções de Rede. Essas funções de rede são, via de regra, implementadas por equipamentos dedicados, com hardware e software proprietários, conhecidos como *appliances*. De acordo com o ETSI, além do alto custo de aquisição destes equipamentos, as tecnologias utilizadas impedem a evolução e/ou modificação destes para implementar novas ideias e tecnologias experimentais. Do ponto de vista do gerenciamento, a tarefa é por vezes realizada apenas no local ou via console, além de exigir modificações nas conexões físicas em certas circunstâncias. Finalmente, o dimensionamento destes é realizado considerando a demanda máxima, gerando desperdícios na aquisição dos equipamentos e no consumo de energia. Quando esta demanda máxima é superada, novos equipamentos devem ser adquiridos, configurados e instalados no local, gerando um tempo de solução para o problema da demanda que pode compreender algo entre horas ou até dias.

Considerando este cenário e com as novas exigências e paradigmas como a Computação em Nuvem, o ETSI propôs em 2012 o conceito de NFV e, na sequência, um conjunto de casos de uso [ETSI 2013] onde a tecnologia pode ser aplicada. A Figura 1 apresenta o conceito básico que apoia a tecnologia NFV, ou seja, a substituição de equipamentos dedicados por versões virtuais suportadas por servidores e equipamentos de rede *commodities* utilizando técnicas de virtualização.

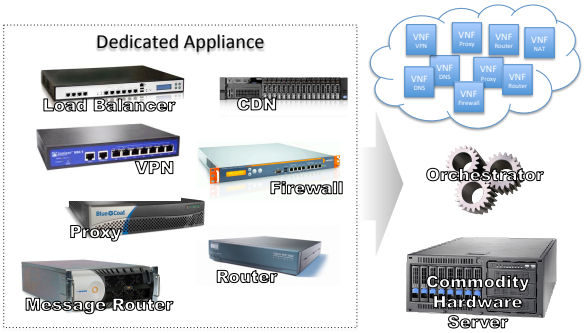


Figura 1: Conceito Básico de NFV.

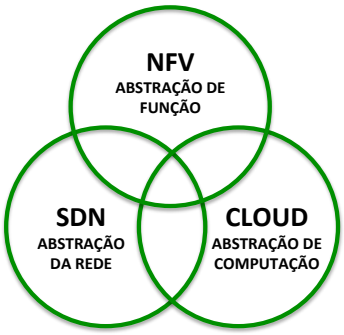


Figura 2: Relacionamento entre NFV, SDN e Computação em Nuvem.

A visão apresentada pelo ETSI mostra claramente uma relação, não dependente, entre NFV, SDN e Computação em Nuvem. Em [Mijumbi et al. 2015, Open Networking Foundation 2015], essa relação apresentada na Figura 2 mostra o benefício mútuo presente na intersecção entre NFV e SDN, promovendo a automação, isolação e agilidade entre NFV e Computação em Nuvem na orquestração, elasticidade e provimento de recursos. A intersecção das três tecnologias permite a implementação do conceito de Infraestrutura como Serviço (IaaS) de forma ampla e eficiente.

3.2 Técnicas de Virtualização

O elemento fundamental de NFV é a técnica de virtualização e esta pode influenciar diretamente na performance e versatilidade da implementação das Funções de Rede

Virtualizadas (VNF). Entre as técnicas disponíveis, este trabalho avalia a virtualização total, a paravirtualização e os containers Linux.

Na virtualização total, é fornecido pelo sistema hospedeiro uma camada completa de abstração, onde todo o hardware é virtualizado – neste tipo de virtualização o sistema operacional(SO) hóspede não detecta que está sendo virtualizado. A Figura 3(a) apresenta a relação entre o SO hóspede, o hardware virtualizado e o SO hospedeiro.

Já a paravirtualização modifica o SO hóspede, como pode ser observado na Figura 3(b), reduzindo as camadas de abstração até o hardware, com o objetivo de melhorar a performance e o controle do SO hóspede pelo hipervisor – como há uma série de modificações que são realizadas no SO hóspede, diz-se que, neste caso, este consegue detectar que está sendo virtualizado.

O terceiro tipo de virtualização abordado neste trabalho não possui todas as característica que definem uma máquina virtual. Os containers são um método de compartilhamento de máquina, onde o mesmo kernel é utilizado por vários ambientes operacionais, utilizando para isso as características de separação de espaço de nomes e diretório raiz nativas do Linux, como pode ser observado na Figura 3(c). Define-se um sistema de arquivos, partição de memória, usuários, grupos e até adaptadores de rede virtuais individuais, porém sobre o mesmo SO hospedeiro. Este tipo de "virtualização" apresenta características importantes, principalmente no acesso ao hardware que não é virtualizado e, sim, diretamente acessado. Por outro lado, questões de segurança e portabilidade são questionáveis nesta abordagem.

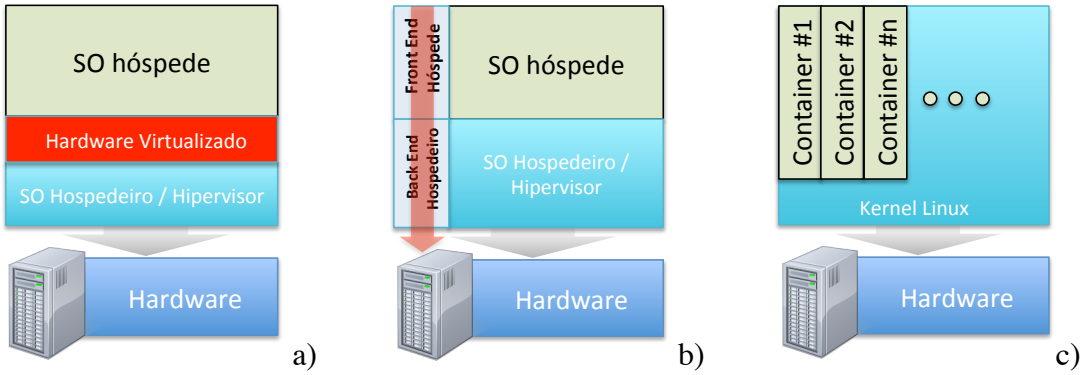


Figura 3: Virtualização Completa, Paravirtualização e Linux Containers.

Há inúmeras implementações destes diferentes métodos de virtualização e alguns virtualizadores também são capazes de utilizar tanto o método de virtualização total como a paravirtualização. Nos experimentos realizados utilizou-se o KVM para virtualização total, o XEN para a paravirtualização e o LXC para containers.

3.3 Switch Virtual – Open vSwitch

O conceito de virtualização prevê a existência de múltiplas máquinas virtuais sob uma única máquina física. Sob esta premissa, a ideia de interconexão destas múltiplas máquinas em uma rede é necessária, ou seja, uma infraestrutura de rede virtual. Além disso, a interconexão com diferentes máquinas reais e com a infraestrutura física também faz-se necessária. Dentre as muitas soluções para esta questão, o projeto Open vSwitch[5] (OVS) mostrou-se eficaz, não apenas por prover virtualmente todas as

[5] http://www.openvswitch.org. Acesso em 01/04/2015.

facilidades e funcionalidades dos switches mais modernos, como também pela possibilidade de utilizar o protocolo OpenFlow [McKeown et al. 2008] para sua operação, promovendo o uso do paradigma SDN. Alia-se a essa característica a adoção do OpenFlow por switches físicos, o que permite a interação direta entre equipamentos físicos e virtuais de forma transparente.

4 Metodologia

Os experimentos realizados consideraram a amostra do número médio de usuários conectados em 117[6] praças da cidade de São Paulo [Ratusznei et al. 2015]. A Figura 4 apresenta a distribuição destes acessos em cada praça utilizada para produzir o tráfego necessário ao experimento, totalizando 3.813 usuários simultaneamente. O tráfego gerado por estes usuários, mostrado na Figura 5, consiste na transferência de arquivos selecionados uniformemente, com tamanhos que seguem uma distribuição Lognormal com média 5Kbytes e variância $1{,}65 \times 10^{-4}$.

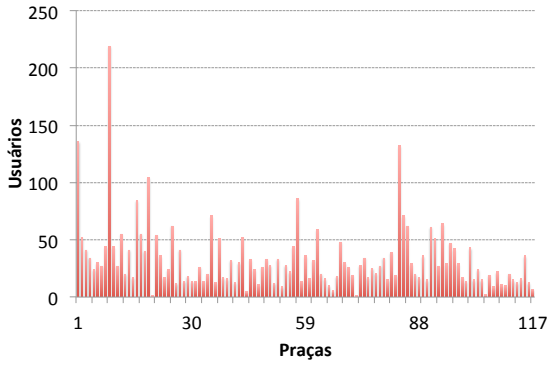


Figura 4: Distribuição de Usuários Entre as Praças.

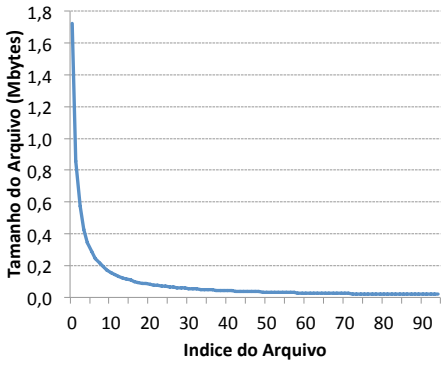


Figura 5: Distribuição de Tamanho dos Arquivos.

Na Figura 6 tem-se o esquema descritivo do experimento, no qual as Praças Digitais, doravante denominadas simplesmente Praças, são representadas com seus respectivos usuários. Em cada Praça há uma infraestrutura básica para prover os serviços de conexão sem fio (Wi-Fi) dos usuários e realizar o encaminhamento do tráfego para as instalações da empresa responsável por tratá-lo e realizar a troca de tráfego com a Internet.

Ainda na Figura 6, tem-se o controlador de elasticidade com NFV[7]. A função deste controlador é monitorar a utilização das VNFs instanciadas para efetuar a tradução dos endereços de rede (NAT), encaminhando o tráfego para aquelas com menor nível de utilização, modificando, para isso, a rota padrão (gateway) das Praças envolvidas na sobrecarga. No caso de sobrecarga do conjunto das VNFs, o controlador cria uma nova instância para dividir o tráfego. Caso o uso das instâncias esteja abaixo do limiar inferior, as instâncias com menor carga são eliminadas e suas Praças dependentes são direcionadas para as VNFs restantes.

[6] O projeto Wi-Fi Livre SP da Prefeitura de São Paulo contempla 120 praças porém os dados de utilização estavam disponíveis em apenas 117 em agosto de 2015.

[7] Código fonte disponível em: https://github.com/heideker/elasticNFV.git

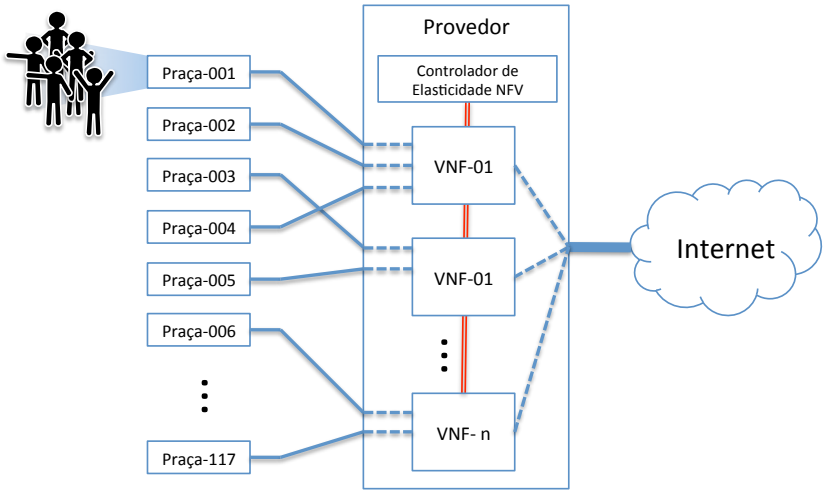


Figura 6: Esquema Descritivo do Experimento.

No início dos experimentos, há apenas uma VNF de NAT instanciada para tratar o tráfego. Como é imposto aos geradores de tráfego um crescimento no número de requisições para forçar o mecanismo de elasticidade a entrar em ação, nos primeiros instantes do experimento esta única VNF é capaz de atender a demanda do sistema. Quando o número de requisições começa a aumentar, o controlador inicia a sua atuação. A Figura 7 apresenta uma amostra da taxa de conexões ao longo do experimento submetida ao sistema.

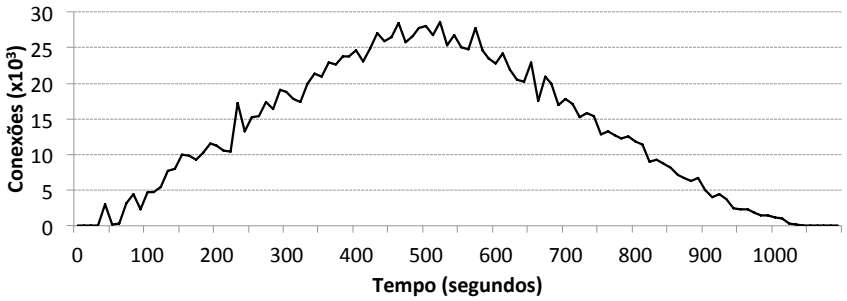


Figura 7: Taxa de Conexões ao Longo de Experimento.

O cenário proposto para o experimento e descrito na Figura 6 é traduzido no esquema lógico da Figura 8. Três servidores físicos são utilizados para produzir o tráfego, onde cada Praça é representada por um container LXC, que, por sua vez, hospeda processos que geram o tráfego equivalente ao número médio de usuários apurados na praça em questão.

Ainda na Figura 8, tem-se o servidor de NFV, que fornece VNFs utilizando uma das três tecnologias abordadas neste trabalho: KVM, XEN ou LXC. A criação das VNFs é orquestrada pelo Controlador NFV que monitora o seu estado, criando novas instâncias, modificando as rotas das Praças para melhor acomodação do tráfego ou dispensando instâncias subutilizadas, de acordo com a política de elasticidade. Finalmente, há no esquema lógico um servidor web responsável por responder ao tráfego gerado pelas Praças. A Tabela 1 mostra a especificação dos equipamentos utilizados nos experimentos.

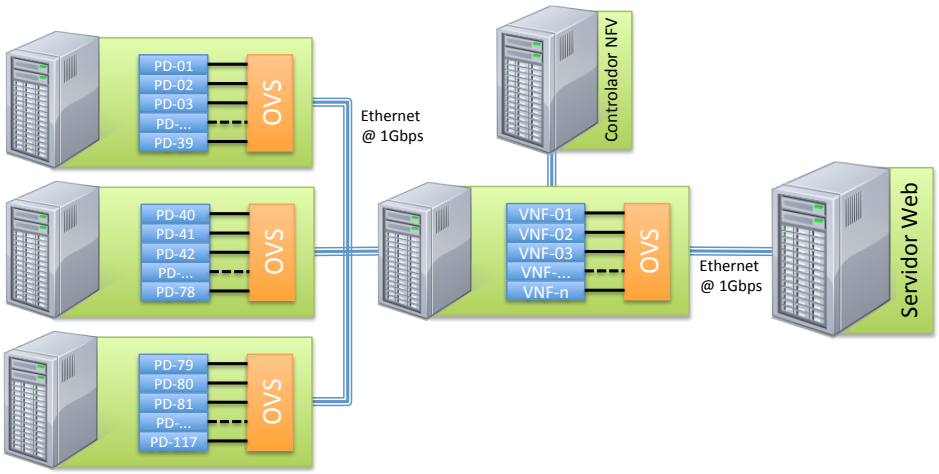


Figura 8: Esquema Lógico do Experimento.

Para realizar os experimentos foi escolhido o NAT como função de rede, por ter características adequadas ao cenário proposto. Com maior ou menor nível de complexidade, todas as funções de rede recebem pacotes, fazem algum tipo de inferência ou modificação e encaminham pacotes para a fonte ou para o destino. No caso do NAT, os pacotes são recebidos, registrados e modificados, encaminhados ao destino e, em seu retorno, novamente tratados.

Tabela 1: Especificação dos Equipamentos Utilizados nos Experimentos.

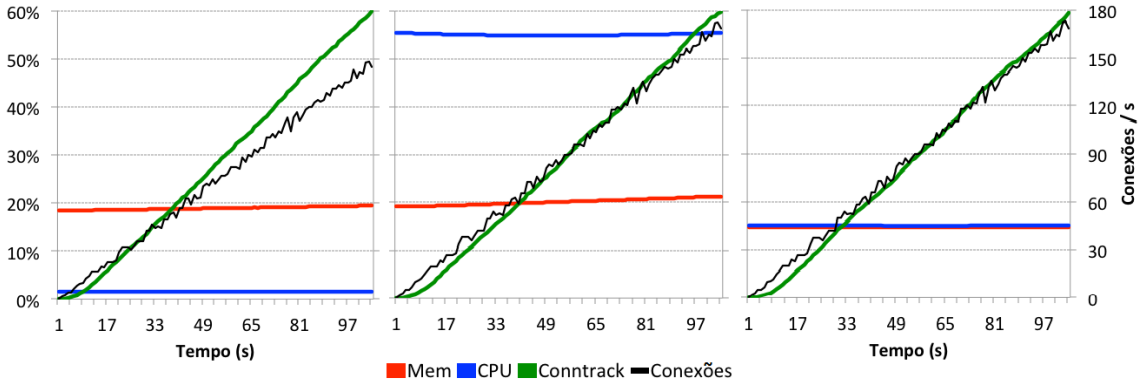
Equipamento	Características
Geradores de Tráfego (3)	Intel® Xeon® E3-1240 V2 @ 3.40GHz 4 núcleos
Servidor de VNFs (3)	Intel® Xeon® E3-1240 V2 @ 3.40GHz 4 núcleos
Controlador de NFV (1)	Intel® Xeon® E5410 @ 2.33GHz 4 núcleos
Servidor Web (1)	Intel® Xeon® E3-1240 V2 @ 3.40GHz 4 núcleos
Switch Físico (1)	48 portas Gigabit Ethernet (IEEE 802.3z 1000Mbits/s)

Para monitorar o estado da VNF, identificando seu nível de utilização, um pequeno processo é executado nesta , respondendo às requisições TCP na porta 5999, informando seu estado atual. O controlador faz o monitoramento das VNFs instanciadas em intervalos regulares de 10 segundos.

No caso da função de rede NAT, em especial na sua implementação no Linux através do IPTables, há uma estrutura de dados que limita o número de conexões simultâneas administráveis pelo NAT – a tabela *conntrack* – que, por sua vez, tem seu tamanho limitado à quantidade de memória disponível na máquina em questão. Para o tamanho das máquinas virtuais utilizadas para hospedar as VNFs foi definido, considerando o tamanho típico de uma instância utilizadas nas plataformas de computação em nuvem, com 512Mbytes de RAM. Para este tamanho de memória, o tamanho da tabela *conntrack* foi definido com 8192 posições, ou seja, quando 8192 conexões simultâneas são rastreadas pela VNF, o monitor reporta ao controlador um nível de 100% de utilização.

Nos experimentos preliminares, as métricas de CPU e de memória também foram avaliadas, porém sem resultados conclusivos. A Figura 9 apresenta a comparação

entre estas métricas, submetendo a VNF a um volume crescente de conexões. Tanto o nível de utilização de CPU como a ocupação de memória apresentaram uma variação insignificante ao longo do experimento, não demostrando correlação direta com o nível de utilização da VNF. No caso da tabela *conntrack*, há uma relação direta com o crescimento do tráfego, mostrando-se como métrica adequada para monitoramento no caso do NAT.

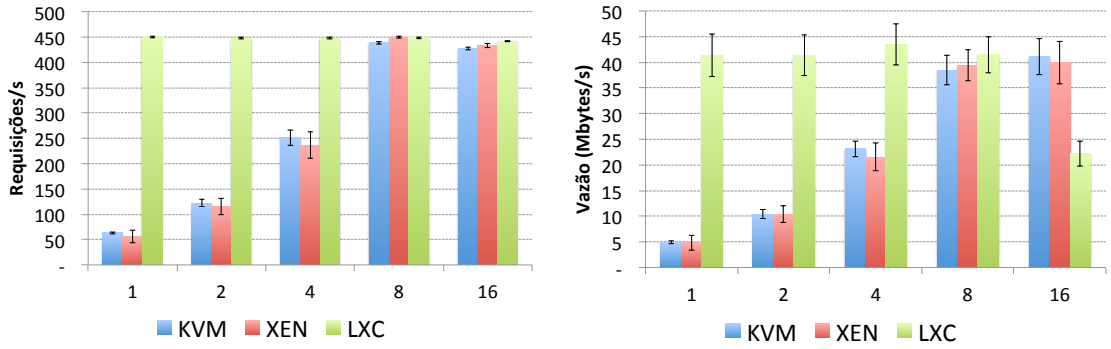


a) Métricas em KVM. b) Métricas em XEN. c) Métricas em LXC.

Figura 9: Comparação de Métricas de Monitoramento

5 Resultados

O primeiro experimento submete o sistema a um tráfego constante, variando o número de VNFs que atende as Praças. Foram realizados experimentos com 1, 2, 4, 8 e 16 VNFs com o objetivo de avaliar a escalabilidade em relação à tecnologia de virtualização. A Figura 10(a) apresenta o número de requisições por segundo admitidas pelo sistema. Apesar do número de conexões admitidas crescer quando o tráfego é dividido entre um maior número de VNFs, observa-se, na Figura 10(b) que há uma diferença expressiva na vazão com o uso de LXC.

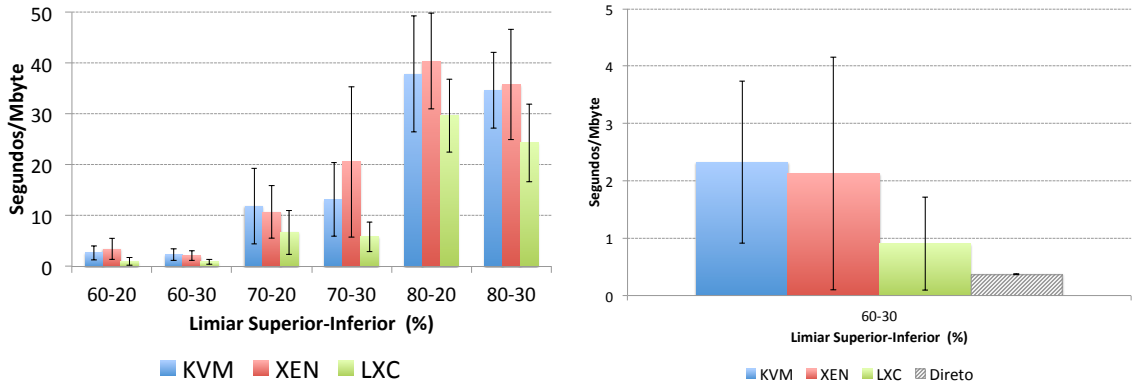


a) Requisições Admitidas pelo Sistema. b) Vazão Média Obtida nas Praças.

Figura 10: Escalabilidade do Serviço.

O segundo experimento considera o tráfego com crescimento constante (Figura 7), utilizando o mecanismo de elasticidade para gerenciar o uso das VNFs. A Figura 11(a) apresenta a taxa de transferência em segundos por megabyte obtida pelo sistema, utilizando diferentes limiares superiores e inferiores para o mecanismo de elasticidade. A avaliação dos resultados obtidos levou à escolha do limiar "60-30", entre os pares experimentados, como adequado à operação do sistema. A Figura 11(b) apresenta com

maior resolução o resultado obtido por este limiar e a sua comparação com a abordagem direta (sem o uso de NFV).



a) Comparação Entre Diferentes Limiares. b) Comparação Entre as Tecnologias.

Figura 11: Taxa de Transferência (s/MB).

Outra métrica explorada neste experimento é o *Round Trip Time* (RTT), medido entre cada Praça e o servidor Web. A Figura 12 apresenta os resultados obtidos e a comparação com a abordagem direta (sem o uso de NFV).

O tempo necessário para a criação de uma nova instância de VNF durante a operação do sistema pode influenciar o desempenho geral do sistema, haja vista que a demora na disponibilização deste novo recurso pode sobrecarregar as VNFs existentes. A Figura 13 apresenta o tempo médio de criação das VNFs durante a execução do experimento com diferentes limiares.

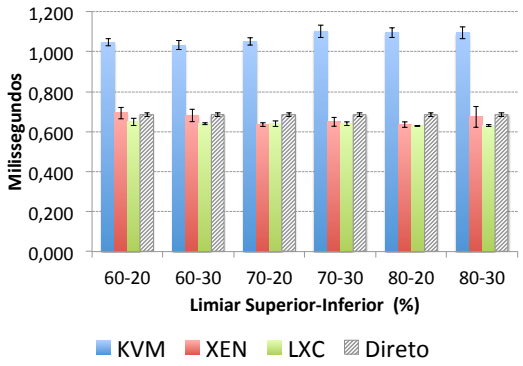


Figura 12: RTT entre Praças e Servidor Web

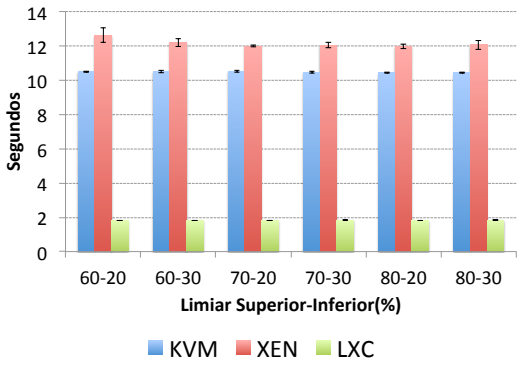


Figura 13: Tempo Médio de Criação de VNF

Outra métrica relacionada diretamente à VNF é a sua carga média, observada durante o experimento, e a influência dos diferentes limiares nesta métrica. A Figura 14 apresenta o resultado obtido para esta métrica. Também relacionado ao uso das VNFs e ao custo de operação do sistema, tem-se a ocupação média de recursos, apresentada na Figura 15. Todos os experimentos foram realizados com 30 repetições com intervalos de confiança de 99%.

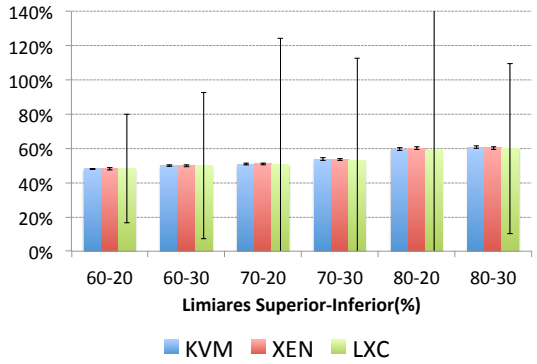


Figura 14: Carga Média das VNFs.

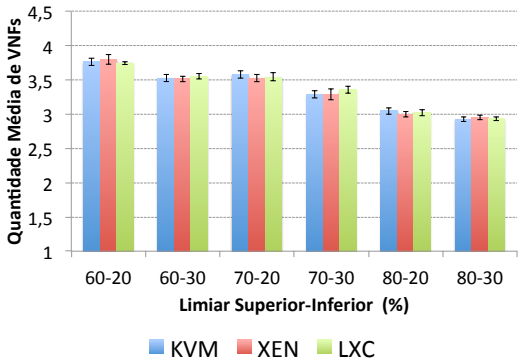


Figura 15: Ocupação Média de VNFs.

As Figuras 16 (a), (b) e (c) apresentam uma série temporal de um dos experimentos com KVM, XEN e LXC, respectivamente, considerando os limiares 60-30. Estas séries apresentam o nível de utilização de cada uma das VNFs instanciadas no decorrer do experimento, assim como a curva de tráfego à qual as VNFs estão submetidas.

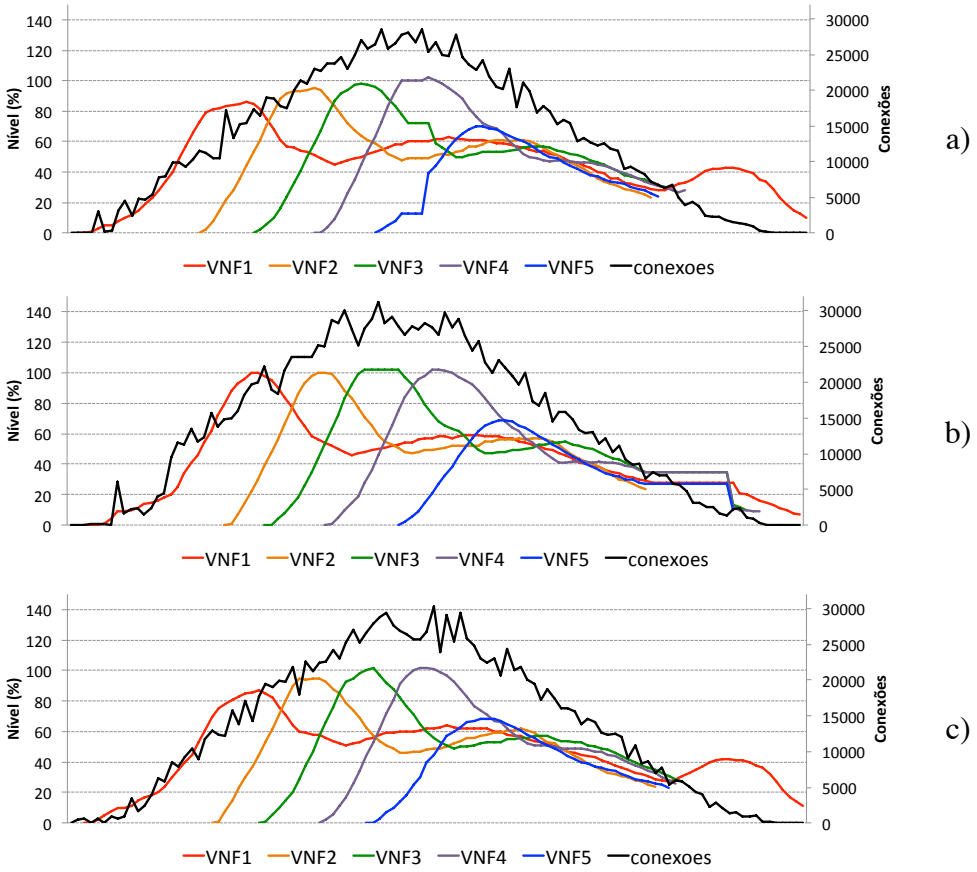


Figura 16: Amostra das Séries Temporais dos Experimentos com KVM(a), XEN(b) e LXC(c).

6 Discussão

A escalabilidade é, sem dúvida, uma das características mais desejadas em ambiente onde os recursos são virtualizados, com a possibilidade de atender diferentes demandas de forma rápida e eficiente. Apesar da tecnologia LXC apresentar um desempenho superior na maior parte das métricas avaliadas, uma característica da tecnologia de virtualização em containers apresenta um obstáculo para sua adoção – é possível observar nos resultados obtidos na Figura 10(b) uma clara redução na vazão quando muitas instâncias de LXC compartilham o mesmo kernel.

No caso específico da função de rede NAT, é possível observar que o fato do kernel da máquina hospedeira ser compartilhado por todos os containers instanciados, é prejudicial quando muitas instâncias dividem estes recursos. Neste caso, a tabela *conntrack* é única, e sua ocupação é disputada pelas diversas VNFs instanciadas – no caso do XEN e KVM, cada máquina virtual possui seu próprio kernel com seus recursos protegidos, ou seja, a soma das tabelas *conntrack* destas múltiplas VNFs é maior que a de um único hospedeiro LXC, mesmo que ele detenha todos os recursos da máquina física. Mesmo que este efeito seja marcante no caso do NAT, muitas funções de rede podem sofrer da mesma anomalia, tendo em vista que a maior parte das manipulações de pacotes ocorrem dentro do próprio kernel do sistema operacional, como *Firewall*, *Traffic shaping*, entre outros.

Já com relação aos resultados apurados no experimento dinâmico, o comportamento obtido pela arquitetura proposta é bastante satisfatório, corroborado pela sobreposição dos intervalos de confiança entre XEN, LXC e a abordagem direta, permitindo o controle da utilização dos recursos pelo ajuste nos limiares e garantindo um nível de serviço comparável à abordagem tradicional, com a vantagem da escalabilidade dinâmica e da aplicação de recursos de infraestrutura sob demanda.

Os resultados obtidos de RTT mostraram uma desvantagem do KVM em relação às tecnologias XEN e LXC, perfeitamente explicável pelo método de virtualização (virtualização total), que aplica mais camadas de abstração entre o SO hóspede e a interface física, implicando um aumento do atraso no processamento dos pacotes.

A avaliação do tempo de criação da VNF mostra uma larga vantagem do LXC, apesar das desvantagens citadas anteriormente. Vale ressaltar que, tanto no XEN, como no KVM, foi utilizada uma máquina virtual sem otimizações, com tamanho típico das máquinas utilizadas em ambientes de computação em nuvem – há trabalhos, como em [Martins et al. 2014], nos quais o foco é a otimização não apenas da máquina virtual como do hipervisor para garantir tempos de instanciação da ordem de 30 milissegundos.

Os resultados obtidos pela avaliação da utilização dos recursos apresentaram valores coerentes com o esperado e equilibrados, independentemente da tecnologia utilizada. Observa-se, na Figura 14 que os intervalos de confiança obtidos pelo experimento utilizando LXC tornaram o resultado inconclusivo, explicável pela característica mencionada do compartilhamento da tabela *conntrack* pelos múltiplos containers, não permitindo o correto monitoramento de seu estado ao longo do tempo.

Finalmente, as séries temporais obtidas apresentaram o comportamento esperado do mecanismo de elasticidade, como a criação das novas VNFs com o crescimento do tráfego, a acomodação de seus níveis com a transferência do tráfego das praças entre as instâncias disponíveis e a dispensa dos recursos ociosos durante a redução do tráfego.

7 Conclusões

O experimento proposto permitiu identificar claramente as vantagens preconizadas pelo ETSI para a tecnologia NFV. Versatilidade, alocação dinâmica de recursos e as consequentes reduções de custo são facilmente observadas neste estudo.

No que tange às tecnologias de virtualização, foi possível observar as vantagens que são apresentadas pela tecnologia de containers, assim como suas desvantagens relacionadas à segurança e escalabilidade. Apesar de haver uma pequena vantagem na adoção da paravirtualização utilizada pelo XEN, esta vantagem em relação ao KVM encontra-se dentro do intervalo de confiança dos experimentos.

A aplicação de NFV como IaaS no cenário proposto das Praças Digitais mostrou-se eficaz e uma alternativa à adoção de *appliances* no tratamento do tráfego, além de permitir a expansão desta infraestrutura de acesso público proposta pela Prefeitura de São Paulo. Além disso, novas funções de rede podem ser implementadas para suprir outras demandas deste cenário, como proxy-cache, DNS, Firewall, entre outros, criando *service chainnings* mais complexos.

Como trabalhos futuros, consideramos que a utilização da tecnologia SDN no mecanismo de elasticidade pode tornar ainda mais versátil a solução apresentada, além do tratamento de VNFs sensíveis ao estado e experimentos utilizando outras tecnologias de virtualização e containers como o VMWare[8] e o Docker[9].

Referências

Bari, Md. F., Shihabur R. C., Reaz A. and Boutaba, R. (2015) "On Orchestrating Virtual Network Functions in NFV". 11th International Conference on Network and Service Management (CNSM). IEEE. p50-56.

Bronstein, Z. and Shraga, E. (2014) "NFV virtualisation of the home environment". Consumer Communications and Networking Conference (CCNC). IEEE. p899-904.

Császár, A., John, W., Kind, M., Meirosu, C., Pongrácz, G., Staessens, D. and Westphal, F. J. (2013). "Unifying cloud and carrier network: Eu fp7 project unify". IEEE/ACM 6th International Conference. IEEE. P452-457.

Demchenko, Y., Filiposka, S., Tuminauskas, R., Mishev, A., Baumann, K., Regvart, D., and Breach, T. (2015). "Enabling Automated Network Services Provisioning for Cloud Based Applications Using the Zero Touch Provisioning". 5th Workshop on Network Infrastructure Services. IEEE.

ETSI. (2013) "Network Functions Virtualisation (NFV): Use Cases". IEEE Network. Disponível em: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/ 01.01.01_60 /gs_NFV001v010101p.pdf. Acesso em: 18/02/2015.

ETSI, Chiosi, M., Clarke, D., Willis, P., Reid, A., Feger, J., Bugenhagen, M. et. al. (2012). "Network Functions Virtualisation". ETSI. Disponínel em: https://portal.etsi.org/nfv/nfv_white_paper.pdf. Acesso em: 04/01/2015.

[8] http://www.vmware.com. Acesso em: 16/12/2015.

[9] https://www.docker.com. Acesso em: 10/11/2015.

Gember-Jacobson, A., Viswanathan, R., Prakash, C., Grandl, R., Khalid, J., Das, S. and Akella, A. (2015). "OpenNF: Enabling innovation in network function control". ACM SIGCOMM Computer Communication Review. Vol 44. ACM. p163-174.

Heideker, A. and Kamienski, C. (2015). "Funções de Rede Virtualizadas em Plataforma de Computação em Nuvem para Cidades Inteligentes". In XIII Workshop em Clouds e Aplicações – WCGA. SBC. p43–56.

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Huici, R. and Bifulco, F. (2014). "ClickOS and the Art of Network Function Virtualization". 11th USENIX Symposium on Networked Systems Design & Implementation. USENIX. p459–473.

Mckeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J. (2008). "OpenFlow: Enabling Innovation in Campus Networks". ACM SIGCOMM Computer Communication Review. ACM. p69-74.

Mijumbi, R., Serrat, J., Gorricho, J. L., Bouten, N., Turck, F. and Boutaba, R. (2015). "Network Function Virtualization: State-of-the-art and Research Challenges". IEEE Communications Surveys & Tutorials. IEEE. p236-262.

Open Networking Foundation. (2012). "Software-Defined Networking: The New Norm for Networks". Disponível em: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/onf2015.310_Architectural_comparison.08-2.pdf. Acesso em: 14/11/2015.

Open Networking Foundation. (2015). "Relationship of SDN and NFV". Disponível em: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/onf2015.310_Architectural_comparison.08-2.pdf.Acesso em: 14/11/2015.

Patouni, E., Merentitis, A., Panagiotopoulos, P., Glentis, A. and Alonistioti, N. (2013). "Network Virtualisation Trends: Virtually Anything Is Possible by Connecting the Unconnected". IEEE SDN for Future Networks and Services. IEEE. p1–7.

Ratusznei, J., Silva, W., Pinheiro, N., Melo, R. and Kamienski, C. (2015). "Uma Rede WiFi Aberta de Larga Escala como Infraestrutura para Cidades Inteligentes". 42º SEMISH - Seminário Integrado de Software e Hardware. SBC.

Roth, Z., Goldhamer, M., Chayat, N., Burr, A., Dohler, M., Bartzoudis, N. and Walker, C. (2010). "Vision and architecture supporting wireless GBit/sec/km2 capacity density deployments". Future Network & Mobile Summit. IEEE. p1-7.

Soares, J., Dias, M., Carapinha, J., Parreira, B. and Sargento, S. (2014). "Cloud4NFV: A platform for Virtual Network Functions". IEEE 3rd International Conference on Cloud Networking. p288–293.

Sousa, J. R. B., Sausen, P. S., Lima, a. M. N. and Perkusich. (2007). "Redes de petri híbridas diferenciais: aplicação na modelagem e no gerenciamento dinâmico de energia de redes de sensores sem fio". SBA: Controle & Automação. p278–291.

Stojmenovic, I. and Wen, S. (2014). "The Fog computing paradigm: Scenarios and security issues". Computer Science and Information Systems (FedCSIS). IEEE. p1-8.

Zheng, Y., Capra, L., Wolfson, O. and Yang, H. (2014). "Urban Computing: Concepts, Methodologies, and Applications". ACM Transactions on Intelligent Systems and Technology. ACM. p1–55.

ROUTEOPS: Orquestração de Rotas Centrada na Operação de Sistemas Autônomos

Rafael S. Guimarães[1,2], Magnos Martinello[1], Cristina K. Dominicini[1,2]
Dione S. A. Lima[1], Rodolfo S. Villaça[1], Moisés Renato N. Ribeiro[1] *

[1] Núcleo de Estudos em Redes Definidas por Software (NERDS)
Universidade Federal do Espírito Santo (UFES) – Vitória, ES

[2]Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo (IFES)
Campus Cachoeiro/ES, Serra/ES – Brazil

rafaelg@ifes.edu.br, magnos@inf.ufes.br, cristina.dominicini@ifes.edu.br
dione.sousa@gmail.com, rodolfo.villaca@ufes.br, moises@ele.ufes.br

Abstract. *The Internet is formed by the union of different autonomous systems (ASes), which use the BGP protocol for exchanging routing information. In each AS, the operators make use of archaic and obscure mechanisms for defining intra-AS and inter-AS policies. In this context, this paper presents ROUTEOPS: an architecture centered on ASes operation for orchestrating routes. The proposal is based on the use of a SDN controller with centralized view of the AS that assists in defining more expressive policies and facilitate their implementation. The centralized vision is guaranteed by integrating ROUTEOPS with eBGP announcements from neighboring ASes; while control and greater expressiveness of routing policies are achieved with OpenFlow and the centralization of iBGP announcements (intra-AS). The proposed architecture is implemented in Mininet and several experiments were executed as proof of concept, in order to demonstrate its feasibility and benefits.*

Resumo. *A Internet é formada pela união de diversos sistemas autônomos (ASes), que utilizam o protocolo BGP para troca de informações de rotas. Em cada AS, seus operadores fazem uso de mecanismos arcaicos e obscuros na definição de políticas intra-AS e inter-AS. Nesse contexto, este artigo apresenta a ROUTEOPS: uma arquitetura para orquestração de rotas centrada na operação de ASes. A proposta se baseia na utilização de um controlador SDN com visão centralizada do AS que auxilia na definição de políticas mais expressivas e facilita a aplicação das mesmas. A visão centralizada é garantida pela integração da ROUTEOPS com os anúncios eBGP provenientes dos ASes vizinhos; o controle e a maior expressividade das políticas de roteamento são alcançados com o uso do OpenFlow e da centralização dos anúncios iBGP (intra-AS). A arquitetura proposta é implementada no Mininet e vários experimentos foram executados como prova de conceito, visando demonstrar sua viabilidade e as vantagens de sua utilização.*

*Este trabalho tem recebido financiamento do projeto Horizon 2020 da União Européia para pesquisa, desenvolvimento tecnológico e demonstração sob no. 688941 (FUTEBOL), assim como do Ministério Brasileiro da Ciência, Tecnologia e Inovação (MCTI) por meio da RNP e do CTIC. Além disso, gostaríamos de agradecer o financiamento do CNPq sob no. 456143/2014-9 e 449369/20145, e da FAPES sob no. 524/2015.

1. Introdução

A Internet é formada pela união de aproximadamente 12000 Sistemas Autônomos (ASes, *Autonomous Systems*) independentes. Nesta arquitetura, cada AS possui suas próprias políticas, que são aplicadas à rede mediante distribuição de prefixos IP *inter-AS* usando o protocolo BGP (*Border Gateway Protocol*). Embora o BGP seja o principal protocolo existente para se anunciar rotas entre os sistemas autônomos, a não existência de uma padronização na definição de políticas de cooperação entre os AS (*inter-AS*) dificulta a sua operação interna (*intra-AS*).

As políticas de roteamento *inter-AS* e *intra-AS* precisam ser frequentemente modificadas em função de informações espaciais ("de onde vêm os dados?" ou "para onde eles vão?"), temporais ("quando?") e dos acordos existentes entre os AS (bilaterais ou multilaterais) [Akashi et al. 2006]. Por ser pouco flexível e de difícil gerenciamento, o modelo atual de cooperação faz com que os operadores de rede acabem tendo que usar mecanismos arcaicos e obscuros, tais como a aplicação de filtros baseados somente em prefixos IP, para modificar suas políticas de divulgação. Em geral esses filtros possuem pouca expressividade, são de difícil compreensão e precisam ser corretamente configurados em cada roteador envolvido na política que se quer expressar.

Por geralmente operarem um AS, os Provedores de Serviço de Internet (ISP, *Internet Service Provider*) são as principais entidades interessadas no desenvolvimento de soluções que facilitem a operação de redes baseadas no protocolo BGP. Neles a arquitetura de rede geralmente é composta tanto de arranjos *inter-AS*, com a utilização de eBGP (*External Border Gateway Protocol*), quanto de arranjos *intra-AS*, com a utilização do iBGP (*Internal Border Gateway Protocol*). O iBGP é usado para disseminar, no interior de um AS, informações de rotas aprendidas externamente por meio do eBGP. Além dos problemas citados anteriormente referentes à pouca expressividade das regras e à obscuridade na definição das políticas, por ser um protocolo baseado em um algoritmo de vetor de distância a divulgação de rotas somente entre vizinhos iBGP limita a visão geral dos roteadores. Isso afeta, por exemplo, a visualização de todas as possíveis rotas para um determinado destino a partir de um ponto no interior do AS. Além disso, o encaminhamento sempre é baseado no prefixo IP de destino, limitando a sua expressividade e dificultando a criação de regras específicas para determinados tipos de serviço (vídeo sob demanda ou armazenamento na nuvem), por exemplo.

Este artigo defende a ideia de que uma visão centralizada dos anúncios provenientes dos ASes vizinhos, o controle centralizado dos roteadores no interior do AS e uma maior expressividade na criação das regras de encaminhamento, são elementos centrais para avançar na orquestração de rotas na perspectiva de um AS. Nesse contexto, a separação entre as funções de encaminhamento e controle, existente nas Redes Definidas por Software (SDN, *Software Defined Networks*), oferece aos pesquisadores e operadores de rede uma excelente oportunidade para contribuir na definição e implementação das políticas de roteamento [Rothenberg et al. 2012, Gupta et al. 2014] nos ISPs.

Desta forma, este artigo apresenta a ROUTEOPS: uma arquitetura inovadora para orquestração de rotas centrada na operação de sistemas autônomos. Contrariamente à expressividade limitada presente nas arquiteturas tradicionais de implantação do protocolo BGP, a ROUTEOPS avança o estado da arte i) ao oferecer expressividade mais clara e ampla no controle e divulgação interna dos anúncios de rotas recebidos pelo AS; ii) ao

ampliar a granularidade do controle das políticas de roteamento *intra-AS*, não existente nas formas atuais de implantação de roteamento dinâmico. Por meio de um controlador SDN com uma visão logicamente centralizada da infra-estrutura interna do AS, pode-se auxiliar na definição de políticas mais expressivas e facilitar a aplicação dessas políticas no interior do AS. A visão externa (*inter-AS*) é garantida pela integração da ROUTEOPS com os anúncios eBGP provenientes dos ASes vizinhos. O uso de iBGP e eBGP garante a interoperabilidade da arquitetura ROUTEOPS com roteadores legados e com os AS vizinhos que não fazem uso da ROUTEOPS.

A arquitetura ROUTEOPS é implementada em ambiente emulado baseado no Mininet[1] e vários experimentos foram executados como prova de conceito, visando demonstrar a viabilidade de implementação da arquitetura proposta e as vantagens de sua utilização em um AS. O artigo é estruturado da seguinte forma: Na Seção 2, são apresentadas as características dos anúncios *intra-AS* e aborda os trabalhos relacionados à proposta. A Seção 3 detalha as características da arquitetura ROUTEOPS. A Seção 4 apresenta os resultados como prova de conceito. A Seção 5 conclui o artigo e apresenta os trabalhos futuros.

2. Roteamento BGP e Trabalhos Relacionados

Em um ISP que opera um AS, o protocolo BGP é usado para descobrir rotas a serem percorridas por um pacote para um determinado destino.O iBGP é usado como mecanismo de distribuição das rotas, recebidas via anúncios externos, para o interior do AS. Uma diferença crucial entre eBGP e iBGP, basicamente, é o uso de mecanismos diferentes para a prevenção de "*looping*" no anúncio de suas rotas. Neste caso, o eBGP simplesmente olha para o atributo AS_PATH, que contém a lista de ASes e compara com o ASN (*Autonomous System Number*) de seus vizinhos. O iBGP, por sua vez, utiliza outros meios para lidar com este problema, uma vez que todos os seus vizinhos possuem o mesmo ASN. Contudo, as sessões iBGP podem ser estabelecidas de duas maneiras: *full-mesh* ou centralizados em um único roteador.

A utilização de conexões *full-mesh* no iBGP não é escalável, pois o número de sessões BGP entre os roteadores será de $\frac{\alpha(\alpha-1)}{2}$, sendo α o número de roteadores no ISP. Uma alternativa para este tipo de conexão é a utilização de Reflexão de Rotas (*Route Reflection*) [Society 2006], uma abordagem que habilita um determinado roteador (*Route Reflector*) a atuar como uma espécie de concentrador de rotas, repassando o anúncio de rotas recebidas por ele para outros pares de roteadores iBGP vizinhos. Para a utilização de *Route Reflection*, os operadores de rede devem efetuar configurações adicionais no *Route Reflector* visando delimitar o escopo da reflexão, evitando problemas de "*looping*" nos anúncios para os seus vizinhos. Por outro lado, a reflexão de rotas torna a visão mais restrita em relação aos anúncios de todos os ASes, uma vez que serão refletidas somente as melhores rotas na visão de um determinado roteador (*Route Reflector*).

Em resumo, o uso do iBGP impõe fortes restrições na orquestração de rotas na rede interna de um ISP devido a dois fatores principais: i) a inviabilidade de ligações *full-mesh* entre os roteadores de um AS de médio ou grande porte, o que daria uma maior visão da infraestrutura da rede e; ii) as limitações do mecanismo de reflexão de rotas usando iBGP, que impede a orquestração de rotas com critérios customizados de acordo com a

[1] http://mininet.org/

visão do operador da rede. Nesse contexto, a seguir serão apresentados alguns trabalhos importantes relacionados à ROUTEOPS.

O RCP [Feamster et al. 2004] é pioneiro na criação de uma visão logicamente centralizada do sistema de roteamento, personalizando a distribuição interna dos anúncios com a substituição do método de reflexão de rotas do iBGP. Além disso, a abordagem do RCP simplifica a configuração e seleção da melhor rota: ao invés de utilizar-se dos atributos de seleção do BGP, propõe uma nova camada de interação *inter-AS*. A ROUTEOPS se inspira no RCP no tratamento do roteamento *intra-AS*, substituindo o método convencional de reflexão de rotas pela centralização do controle interno dos anúncios. A utilização, pela ROUTEOPS, de um controlador SDN baseado no protocolo OpenFlow traz uma maior granularidade na configuração do encaminhamento de pacotes e habilita a engenharia de tráfego em decorrência da interação direta com o plano de dados. Mais especificamente, a ROUTEOPS se diferencia do RCP principalmente por não criar uma camada de comunicação *inter-AS*, respeitando a autonomia dos ASes na divulgação de prefixos de rede.

O SDX [Gupta et al. 2014] foi utilizado como base para a criação da ROUTEOPS e também propõe a existência de um controle centralizado, porém com visão completa do encaminhamento *inter-AS*. Para isso, o SDX também faz uso do protocolo OpenFlow para contrapor as principais limitações do roteamento BGP, que são: i) encaminhamento baseado apenas no endereço de destino; ii) influencia apenas sobre seus vizinhos e; iii) expressão indireta das políticas de roteamento. Assim como na ROUTEOPS, o uso do OpenFlow garante uma maior expressividade na criação e manipulação das políticas de cada participante da rede. Entretanto, o SDX delimita seu escopo de controle aos IXPs (*Internet Exchange Points*) e comunicação *inter-AS*, enquanto a *ROUTEOPS* se diferencia por ter como foco os sistemas autônomos, tratando o anúncio de rotas *intra-AS* com uso do iBGP como mecanismo de transporte.

Na visão de uma estrutura *intra-AS*, o BTSDN [Pingping Lin 2014] utiliza informações obtidas nos roteadores de borda para influenciar o encaminhamento *intra-AS*, utilizando o OpenFlow como elemento de configuração de regras de encaminhamento, assim como a ROUTEOPS. A finalidade principal desse arranjo é obter uma cooperação entre uma rede OpenFlow e os roteadores de borda legados, sem alterá-los, com a criação de regras de fluxo auxiliadas pela visão interna dos anúncios. Através da utilização do iBGP, o controlador não tem apenas informações de anúncios *intra-AS*, mas também informações *inter-AS* de forma indireta. A principal diferença entre a arquitetura *ROUTEOPS* e o BTSDN, é que a primeira trabalha de forma ativa na divulgação dos anúncios *intra-AS*, permitindo a divulgação de anúncios personalizados para determinados roteadores, enquanto a última apenas faz o mapeamento das rotas em regras de encaminhamento OpenFlow de forma passiva.

O RouteFlow [Nascimento et al. 2011] é uma abordagem que faz uso da separação entre plano de dados e controle por meio do protocolo OpenFlow. O RouteFlow, assim como o BTSDN, também faz um mapeamento de rotas em regras de encaminhamento OpenFlow, utilizando-se de máquinas virtuais e *switches OpenFlow* para implementação do plano de dados. No Routeflow, o *switch* terá as mesmas funções de um roteador convencional, através da realização do encaminhamento usando reescrita do endereço de MAC de destino. Com isso, o RouteFlow habilita a experimentação de diferentes for-

mas de encaminhamento no plano de dados associado aos anúncios BGP. A principal diferenciação para a ROUTEOPS é que no RouteFlow existe uma rede sobreposta de roteadores virtuais mapeados em *switches* OpenFlow que realizarão o encaminhamento dos pacotes e, no caso do ROUTEOPS, o switch Openflow é utilizado apenas para melhorar a granularidade do encaminhamento interno, em conjunto com a visão inter-AS proporcionada pelo BGP.

Após uma breve revisão dos principais trabalhos relacionados é importante reforçar as duas principais contribuições da ROUTEOPS: i) a disponibilização de uma maior expressividade no controle e divulgação interna dos anúncios de rotas recebidos pelos ASes vizinhos; ii) a ampliação da granularidade do controle das políticas de roteamento *intra-AS*, não existente nas formas atuais de implantação de roteamento dinâmico. Como contribuição secundária, mas não menos importante, a proposta da ROUTEOPS também propicia um ambiente adequado para experimentação de soluções de encaminhamento inovadoras no interior dos ASes, com utilização do protocolo OpenFlow e *switches* SDN no núcleo dos ASes e roteadores legados nas bordas.

3. ROUTEOPS

A arquitetura proposta baseia-se na integração, de forma centralizada, entre a arquitetura BGP e o protocolo OpenFlow[McKeown et al. 2008], de forma que a primeira fornece acesso aos anúncios intra e inter-domínios de toda a infra-estrutura de um ISP e o último permite a manipulação do plano de dados. A Figura 1 mostra os módulos da arquitetura responsáveis pela construção e manipulação de políticas de roteamento, que serão descritos a seguir.

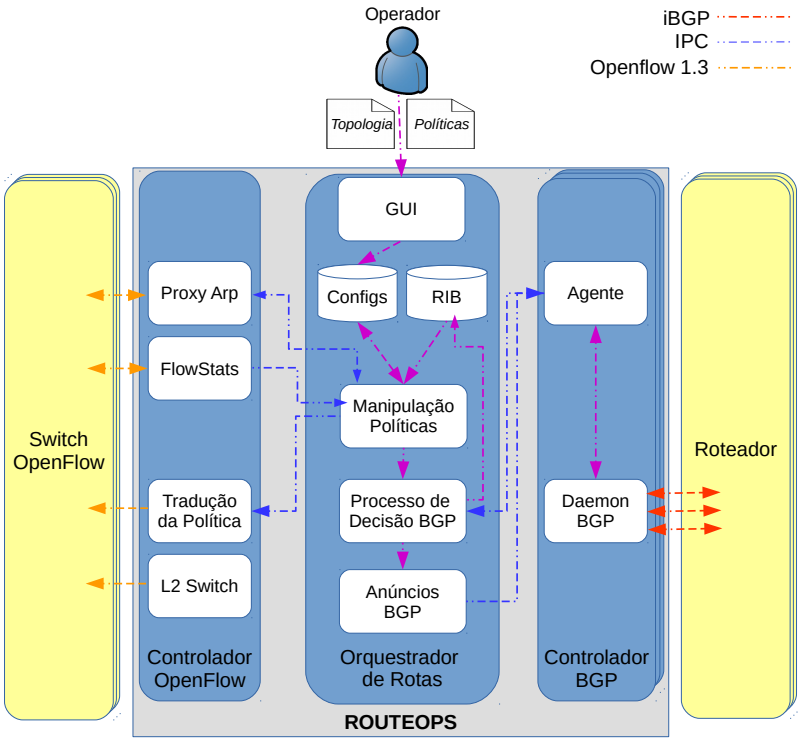
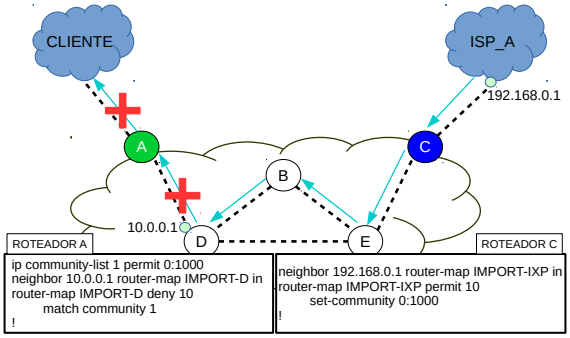


Figura 1. Arquitetura ROUTEOPS

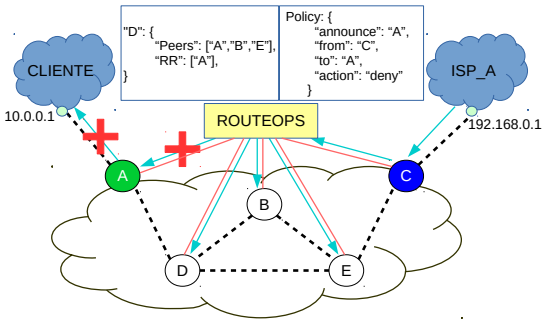
3.1. Orquestrador de Rotas

Suponha que um operador de rede deseja configurar uma política que não aceitará rotas anunciadas pelo *ISP_A*. Na Figura 2(a) o operador de rede deverá configurar em C que as rotas anunciadas por *ISP_A* sejam associadas a uma determinada "*community*". No BGP, "*community*" é um atributo no qual os valores podem estar associados a uma rota, utilizada como uma espécie de marcação para os operadores. No roteador A , baseando-se no valor da "*community*" configurado em C, um filtro deverá ser aplicado. No exemplo, nota-se que uma inconsistência nas configurações dos roteadores invalidará a política de roteamento, ou seja, deverá existir uma sincronia entre os valores de "*community*" definidos no roteador C e A para que este tipo de política funcione. A *configuração topológica* no BGP é consequência das ligações entre os roteadores, que só trocam informações com seus vizinhos.

Na configuração topológica da ROUTEOPS é preciso definir os endereços MAC (*Media Access Control*), IP (*Internet Protocol*), as ligações lógicas entre os roteadores e a política de reflexão das rotas para cada roteador do AS. Para a criação de políticas, o operador poderá criar regras que manipulam o processo de decisão dos anúncios no BGP e regras específicas que diversificam o encaminhamento interno dos pacotes. Por exemplo, na Figura 2(b), para o roteador D foram definidas ligações entre os roteadores (*Peers*) A, B e E e *Router Reflector* (RR) para A, o que significa que A receberá anúncios de B e E mesmo não sendo seus vizinhos.



(a) Arquiteturas Convencionais



(b) ROUTEOPS

Figura 2. Exemplo de expressividade de políticas utilizando método tradicionais em comparação com as políticas definidas no ROUTEOPS

Voltando ao exemplo onde o operador deseja configurar uma política que não aceitará rotas anunciadas pelo *ISP_A*, vamos ilustrar como essa política pode ser expressada de forma simples com a utilização da ROUTEOPS. Na Figura 2(b), descrevemos que os anúncios originados em C e destinados a A serão negados ("action deny").

Como podemos observar, a definição de uma determinada política no modelo tradicional é obscura, conforme observado na Figura 2(a), e a determinação da política é descentralizada com uma visão limitada dos anúncios de todas as rotas. Com a utilização da ROUTEOPS, expressamos a topologia e as políticas diretamente no módulo *Orquestrador de Rotas* através de uma *GUI (Graphical User Interface)* de acesso às configurações e políticas. Desta forma o operador terá um leque de opções de políticas que diversificam o encaminhamento dos anúncios para um determinado roteador, entregando a melhor rota para aquele roteador. Tudo isso em função de uma visão e controle centralizados e a possibilidade de orquestração destes anúncios, com base em critérios definidos pelo operador.

Além disso, o módulo *Orquestrador de Rotas* recebe e envia mensagens para o módulo *Controlador BGP* e para o módulo *Controlador OpenFlow*. Na comunicação com o módulo *Controlador BGP*, o componente *Processo de Decisão BGP* é responsável por receber todos os anúncios e, em seguida, tomar uma decisão sobre qual deles oferece a "melhor" rota para cada destino. Essas melhores rotas são colocados em uma base de dados, denominada de RIB (*Routing Information Base*). No entanto, a definição da "melhor" rota envolve diversos critérios além de observar o caminho mais curto.

O componente *Processo de Decisão BGP*, após definir as melhores rotas, deverá encaminhar os anúncios para o componente *Anúncios BGP*, utilizando, de forma similar, dos critérios de reflexão de rotas da *RFC4456* utilizados no BGP. O componente *Anúncios BGP*, por sua vez, irá encaminhar esses anúncios para o módulo *Controlador BGP* que, por fim, envia os anúncios para os roteadores. A forma como acontecerá a reflexão de rotas deve ser definida e configurada de acordo com as políticas específicas daquele AS. O componente *Manipulação de Políticas*, a partir das políticas pré-definidas pelo operador de rede, influencia no processo de decisão do protocolo BGP. Essa influência acontece por meio da interação com o componente *Processo de Decisão BGP* e *Tradução da Política* do módulo *Controlador OpenFlow*. A interação com o operador de rede é feita por meio de uma interface gráfica, componente *GUI*, que permite a inserção de configurações de topologia e políticas na base de dados do módulo *Orquestrador de Rotas*.

3.2. Controlador BGP

No módulo *Controlador BGP*, o componente *Daemon BGP* é responsável pela conectividade iBGP (Internal BGP) com os roteadores legados. Portanto, este componente será responsável por estabelecer uma sessão entre os pares BGP (*Peers*) e trocar informações sobre atualização de rotas e notificações de erros.

O componente *Agente* é responsável por receber mensagens do componente *Daemon BGP* e enviar mensagens para o módulo *Orquestrador de Rotas* utilizando o formato JSON[2]. Além disso, o componente *Agente* recebe as mensagens do módulo *Orquestrador de Rotas* e as encaminha para o componente *Daemon BGP* em formato adequado.

[2]http://www.json.org

3.3. Controlador OpenFlow

No módulo *Controlador OpenFlow*, responsável por enviar e receber mensagens OpenFlow para um determinado *switch*, o componente *Tradução da Política* é responsável por traduzir as políticas definidas pelo módulo *Orquestrador de Rotas* em regras OpenFlow na versão 1.3. Além disso, o componente *FlowStats*, recebe dos *switches* informações relacionadas aos fluxos que serão entregues ao componente *Manipulação de Políticas* do módulo *Orquestrador de Rotas*. Já o componente *Proxy ARP* responde as requisições *ARP* obedecendo às determinações expressas nas políticas definidas no módulo *Orquestrador de Rotas*.

3.4. Implementação

A implementação do protótipo descrito neste artigo foi dividida em três partes: Controlador OpenFlow, Controlador BGP e Plano de Gerência de Rotas e Políticas (*Router Server*). Cada item foi referenciado como um módulo na arquitetura proposta. A comunicação entre os módulos será realizada através de mensagens IPC (*Inter-Process Communication*), utilizando o banco de dados distribuído NoSQL MongoDB[3] versão 2.6.5. Além disso, essa base de dados NoSQL é usada para implementar uma fila de mensagens JSON entre os módulos, permitindo a comunicação de perda de desempenho e facilitando o gerenciamento de falhas, depuração e monitoramento.

No módulo *Controlador OpenFlow* foi utilizado o controlador Ryu versão 3.23. Nele foram implementados os componentes *Tradução das Políticas*, *L2 Switch*, *FlowStats* e *Proxy ARP*. As principais tarefas associadas a este módulo são: interpretação das políticas, criação de regras de fluxo no plano de dados, respostas ARP_REPLY e coleta de informações de fluxos.

Na implementação do módulo *Orquestrador de Rotas*, foi utilizado a linguagem Python em conjunto com a biblioteca PyMongo[4] para integração com o MongoDB, responsável pela leitura das mensagens recebidas e enviadas para os outros módulos e pela persistência das informações de rotas, políticas e expressões topológicas. Fazem parte deste módulo os componentes *Processo de Decisão BGP*, *Anúncios BGP* e *Manipulação das Políticas*. As principais funcionalidades deste módulo são: armazenamento dos anúncios de rotas e anúncio customizado de rotas com base em uma determinada política.

No módulo *Controlador BGP*, foi utilizado, como Daemon BGP, o *ExaBGP*[5] e, como *Agente*, um software escrito em Python que se comunica diretamente com o *ExaBGP* para habilitar a troca de mensagens com o módulo *Orquestrador de Rotas*.

4. Prova de Conceito

Este seção apresenta o ambiente de experimentação que foi utilizado para a avaliação do protótipo e os resultados obtidos nos experimentos como prova de conceito. Para uma melhor organização da apresentação da prova de conceito, realizou-se a divisão em 3(três) subseções. A primeira (seção 4.1), descreve as características do ambiente de experimentação. Em seguida (seções 4.2 e 4.3), avaliamos o funcionamento do protótipo

[3]https://www.mongodb.org
[4]https://api.mongodb.org/python/
[5]https://github.com/Exa-Networks/exabgp

em determinadas topologias e políticas. Neste caso, apresentamos resultados medindo a vazão de fluxos na visão de operação do AS. Os casos de uso servem para ilustrar a orquestração de rotas da ROUTEOPS.

4.1. Ambiente de Experimentação

Para validar a proposta, utilizamos como ambiente de emulação o Mininet[6] na versão 2.1.0p2 com uma extensão que provê serviços de forma simplificada, denominada de MiniNext[7] na versão 1.10. Esta versão do Mininet provê suporte ao Openflow 1.3, utilizada como base no protótipo. No plano de dados, foi utilizado como máquina de encaminhamento (*Software Switch*), o Open vSwitch[8] na versão 2.3.2. Além disso, todos os enlaces foram configurados com limite de 1Gbps.

Portanto, em uma dada topologia de ISP operando seu próprio AS, cada roteador executa o software *Quagga*[9] na versão 0.99.22.4 para o serviço (daemon) que implementa o protocolo BGP. Para a geração e análise do tráfego durante os experimentos foi utilizada a ferramenta Iperf[10] na versão 3.0.4 com os protocolos TCP e UDP.

4.2. Orquestração de rotas de entrada e saída

O objetivo do primeiro experimento é mostrar uma regra de reflexão de rotas customizadas que impacta na vazão de 2(dois) fluxos de entrada. A topologia utilizada é ilustrada na figura 3, com fluxos TCP com origem no $AS150$ e com destinos para o $AS300$ e para o roteador $R03$ contido no $AS65000$. Utilizamos 3(três) roteadores trocando mensagens iBGP e ligados a um *Switch Openflow* e ao servidor ROUTEOPS, fazendo o papel de um ISP, e 4(quatro) roteadores que trocam mensagens eBGP para realizar troca de anúncio de rotas entre ASes distintos. O $AS300$, exclusivamente, é um cliente, do ISP de exemplo, ligado ao roteador $R02$.

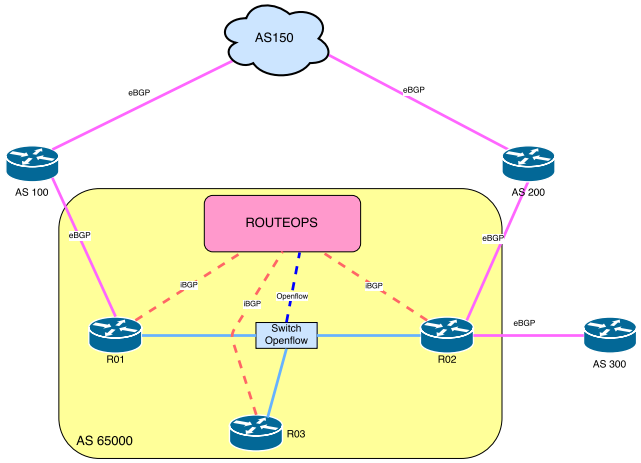


Figura 3. Topologia Cenário 01 e 02

[6]http://mininet.org/
[7]http://mininext.uscnsl.net
[8]http://openvswitch.org/
[9]http://www.nongnu.org/quagga/
[10]https://iperf.fr/

Originalmente, o roteador $R03$ está configurado para refletir suas rotas tanto para $R01$ quanto para $R02$. Estas informações chegam até o $AS150$ via anúncios eBGP que, por sua vez, seleciona o melhor caminho por meio do processo de decisão BGP. Desta forma, os fluxos originados do $AS150$ com destino ao $AS300$(Fluxo 1 na Figura 4) seguia o caminho $AS150$-$AS200$-$R02$-$AS300$. Já os fluxos originados do $AS150$ com destino ao $R03$(Fluxo 2 na Figura 4) seguia o caminho $AS150$-$AS200$-$R02$-$R03$. Assim, como é possível ver na Figura 4 no período de 0s até 200s, a vazão dos fluxos é limitada pela sobrecarga do enlace entre $AS200$ e $R02$. Para resolver tal problema, no instante 200s, o operador insere uma política na *GUI* de acesso ao protótipo para realizar uma reflexão de rotas customizada em que $R03$ reflete suas rotas apenas para $R01$. Desta maneira, conforme ilustrado na Figura 4, o Fluxo 2 com destino para $R03$ passa a ser encaminhado por meio do caminho $AS150$-$AS100$-$R01$-$R03$, liberando o enlace que estava sobrecarregado e aumentando a vazão de ambos os fluxos. Nota-se ainda que houve um tempo de convergência para que cada sistema autônomo passe a interpretar o anúncio das novos anúncios de rotas e a mudança seja efetivada.

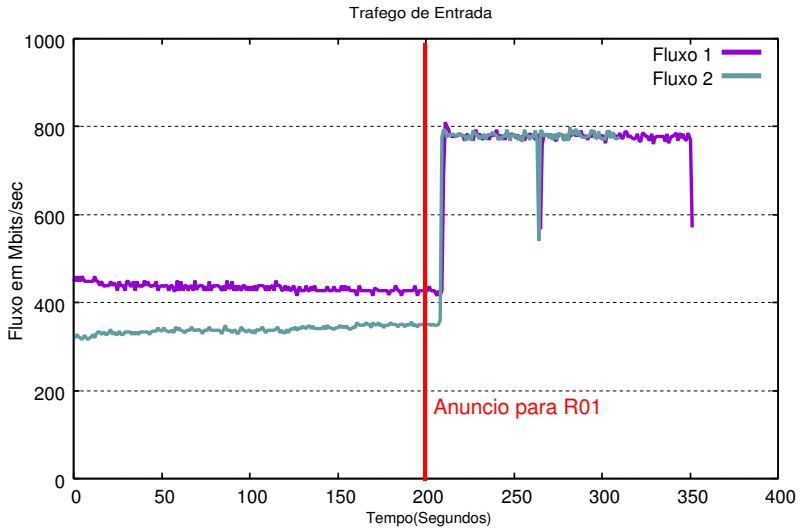


Figura 4. Cenário 01 - Tráfego de entrada

Neste segundo experimento, considera-se como base os fluxos de saída: Fluxo 1 com origem no $AS300$, Fluxo 2 com origem no $R03$ e ambos com destino para o $AS150$. Inicialmente, tanto o roteador $R01$ e $R02$ estão configurados para refletir suas rotas para o $R03$. Desta forma, quando $R03$ deseja encaminhar fluxo para o destino $AS150$ ele utiliza o $R02$ em função do processo de decisão BGP. De forma similar, o $AS300$ quanto deseja encaminhar fluxos para o $AS150$ também escolherá a rota que passe por $R02$. Como consequência, a Figura 5 mostra que durante o período 0s ao 57s a vazão esta limitada devido a sobrecarga do enlace entre $AS200$ e $R02$.

Para resolver tal problema, no instante 57s, o operador insere uma política na *GUI* de acesso ao protótipo para realizar uma reflexão de rotas customizada onde apenas rotas oriundas de $R01$ sejam refletidas para $R03$. Desta maneira, conforme ilustrado na Figura 5, o Fluxo 2 com origem para $R03$ passa a ser encaminhado por meio do caminho $R03$-$R01$-$AS100$-$AS150$, liberando o enlace que estava sobrecarregado e aumentando a vazão de ambos os fluxos. Nota-se ainda que houve um tempo de convergência menor do que o

cenário anterior, já que a mudança do anúncio precisa ser aplicada apenas em $R03$.

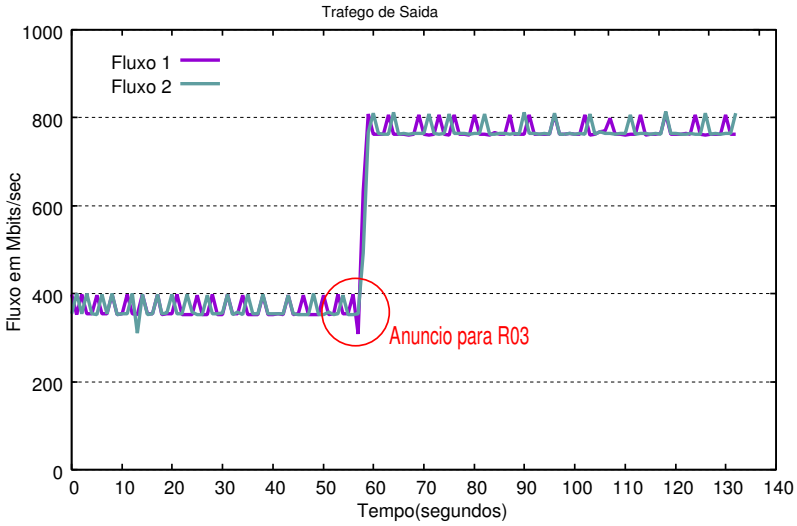


Figura 5. Cenário 02 - Tráfego de saída

4.3. Orquestração de anúncios a partir da monitoração de fluxos

Serviços de vídeo com necessidade de uma alta largura de banda como o YouTube e Netflix, oferecem uma grande demanda de tráfego em relação ao volume global de tráfego, de modo que os ISP estão cada vez mais interessados em trocar tráfego com as aplicações específicas (CDNs) observando sempre o Custo X Benefício para se conseguir realizar estes acordos. De fato, o BGP não torna esta política simples de ser implementada. Um ISP pode determinar o encaminhamento de um determinado fluxo em detrimento da classificação dos serviços atrelados. Para isso, ele deverá identificar o tráfego relevante e efetuar e redirecionar o tráfego para um caminho especial. Por exemplo, um ISP necessitará configurar em seus roteadores de borda diversas tabelas de roteamento que serão utilizadas para uma determinada aplicação e outra tabela de roteamento para as demais aplicações. Este tipo de abordagem cria tabelas com rotas adicionais que sobrecarregam ainda mais a tabela de roteamento e ainda acrescentam uma obscuridade na configuração e classificação dos serviços[Gupta et al. 2014].

Neste experimento é proposta uma solução para esse problema e analisados os impactos de vazão quando um anúncio é feito de forma diversificada por fluxos de cada serviço. A topologia do cenário deste experimento é ilustrada pela Figura 6. Utilizamos 4(quatro) roteadores trocando mensagens iBGP e ligados a um *Switch Openflow* e ao servidor ROUTEOPS, fazendo o papel de um ISP, e 4(quatro) roteadores que trocam mensagens eBGP para realizar troca de anúncio de rotas entre ASes distintos. O $AS300$, exclusivamente, é um cliente ligado aos roteadores $R03$ e $R04$.

Neste cenário, foram configurados dois fluxos com serviços distintos, ambos com origem no $AS150$ e destino para o $AS300$, que competem pelos recursos do meio: o Fluxo 1 do Serviço 1 utiliza o protocolo UDP de forma contínua tentando consumir toda a banda disponível e o Fluxo 2 do Serviço 2 utiliza o protocolo TCP com rajadas de transmissão. Inicialmente, o roteador $AS300$ tem sessões eBGP estabelecidas com os roteadores $R03$ e $R04$, não existe políticas específicas para os anúncios e os dois fluxos são encaminhados

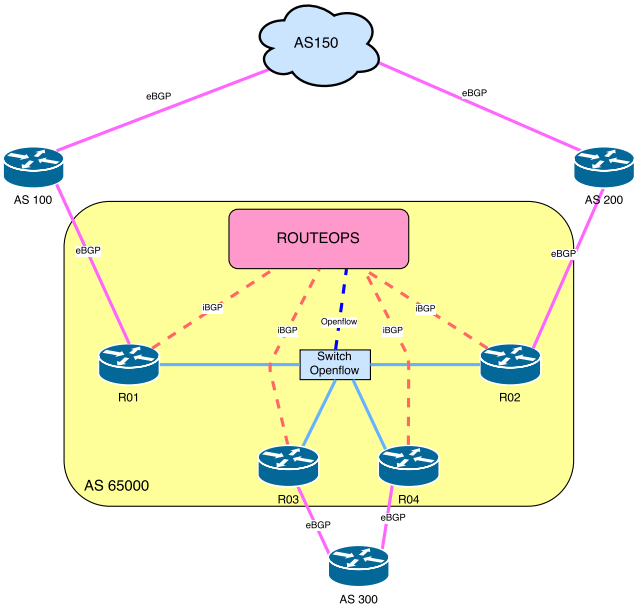


Figura 6. Topologia Cenário 03

pelo mesmo caminho($AS150$-$AS100$-$R01$-$R03$-$AS300$) porém para prefixos distintos do $AS300$.

Com base em um SLA (*Service Level Agreement*) para o cliente do $AS300$, em que será priorizado o QoS (*Quality of Service*) para o Serviço 1, a política determina como ação um anúncio diversificado na infra-estrutura do ISP caso o serviço em questão não consiga garantir uma determinada vazão mínima para os prefixos anunciados pelo $AS300$. No instante anterior a 100s, a vazão do Fluxo 1 diminui em função da competição pelos recursos de rede com o Fluxo 2, conforme ilustrado na Figura 7. Então, o operador de rede detecta que o SLA do Serviço 1 não está sendo cumprido e, após o instante 100s, informa uma política em que determina um valor mínimo de vazão para o Fluxo 1, que neste caso foi de 512Mbps, e uma ação de diversificação de anúncio para o roteador $R02$, onde o roteador $R04$ será responsável por encaminhar os fluxos relacionados aos prefixos do Serviço 1 para o $AS300$. Não alterou-se nenhuma política em relação ao Serviço 2 de forma que o Fluxo 2 continua a seguir pelo caminho $AS150 - AS100 - R01 - R03 - AS300$. Já o Fluxo 1 do Serviço 1, passa a seguir o caminho $AS150 - AS100 - R02 - R04 - AS300$. A partir disso, observamos que além de obtermos uma melhor vazão para o Fluxo 1, melhoramos também a vazão do Fluxo 2. Desta maneira, mostra-se que observar a vazão de um serviço associado a um determinado prefixo, e realizar uma anúncio diversificado em função dessa observação, garante uma engenharia de tráfego mais expressiva dentro da infraestrutura de um ISP.

Para que essa política seja aplicada, é adicionada uma regra específica de encaminhamento no *Switch Openflow* que conta o número de bytes dos fluxos que correspondem ao padrão: Endereços IPs de Origem $AS150$, Endereços IPs de destinos para $AS300$, porta de origem e protocolo de transporte. O intervalo de amostragem será de 10s. Com essa informação, o controlador terá uma amostra da vazão do Serviço 1 entre os ASes $AS150$ e $AS300$. O tempo de coleta de informações de cada fluxo fica condicionado a não deteriorar o desempenho do plano de dados e do plano de controle [Curtis et al. 2011].

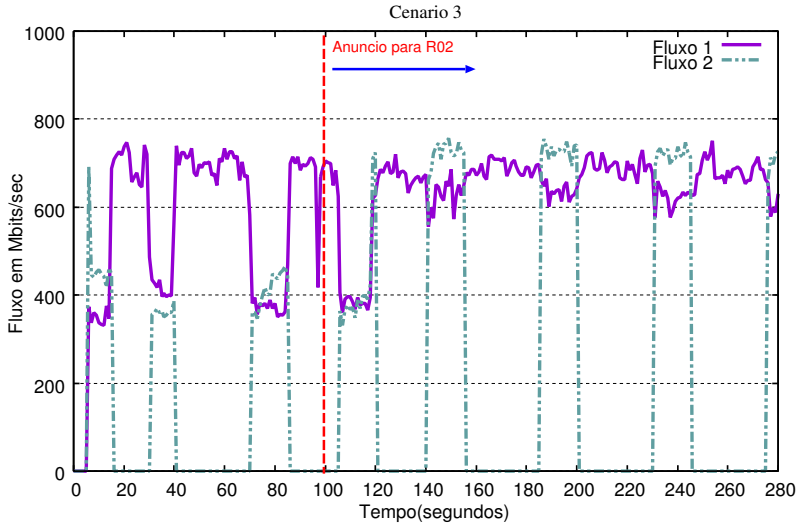


Figura 7. Cenário 03 - Anúncio diversificado com base na análise dos fluxos

5. Conclusões e Trabalhos Futuros

As políticas de roteamento em sistemas autônomos (ASes) são ditadas essencialmente pelo protocolo BGP. O BGP faz uso de mecanismos, tais como a aplicação de filtros baseados somente em prefixos IP, que possuem i) pouca expressividade (arcaicos), ii) são de difícil compreensão (obscuros) e iii) precisam ser corretamente configurados em cada roteador envolvido para que a política seja aplicada na infraestrutura de rede do sistema autônomo (tempo longo de convergência).

Neste contexto, este trabalhado apresenta a ROUTEOPS que é uma arquitetura para orquestração de rotas centrada na operação de sistemas autônomos. ROUTEOPS apoia-se no conceito de SDN com base na visão centralizada provida por um controlador que facilita definir políticas de modo mais claro, garantindo aplicação dessas políticas de modo ágil na infraestrutura interna do AS. A orquestração de rotas é centrada na operação do AS pela integração da ROUTEOPS com os anúncios eBGP provenientes dos ASes vizinhos. Tanto o controle da infra-estrutura interna do AS, quanto a maior expressividade na aplicação das políticas de roteamento são alcançados com o uso de OpenFlow. O protocolo OpenFlow define como será o encaminhamento interno das mensagens iBGP para os roteadores no interior do AS. O uso de iBGP e eBGP garante a interoperabilidade da arquitetura ROUTEOPS com roteadores legados e com os AS vizinhos que não fazem uso da ROUTEOPS.

Como prova de princípio, ROUTEOPS foi implementada em um ambiente de emulação de redes (Mininet). Experimentos demonstraram que ROUTEOPS permite a operação de rotas no AS em tráfegos de entrada e saída no AS que levou a um uso mais equilibrado da infraestrutura do AS. Houve também redução no tempo de convergência para aplicação da política de roteamento intra-AS.

Como trabalhos futuros, pretende-se utilizar a abordagem do trabalho [Alan Chang et al. 2015], no qual realiza uma tratamento misto do encaminhamento através de roteadores convencionais e um *Switch Openflow*, que mostra um melhor desempenho no tempo de convergência do encaminhamento interno. Como outro trabalho

futuro, pretende-se utilizar uma forma de redução na tabela de roteamento proposto pelo trabalho [Luiz Fernando T. de Farias 2014], que permite uma redução das tabelas de encaminhamento (FIB) nos roteadores internos.

Referências

Akashi, O., Fukuda, K., Hirotsu, T., and Sugawara, T. (2006). Policy-based bgp control architecture for autonomous routing management. In *Proceedings of the 2006 SIGCOMM Workshop on Internet Network Management*, INM '06, pages 77–82, New York, NY, USA. ACM.

Alan Chang, M., Holterbach, T., Happe, M., and Vanbever, L. (2015). Supercharge me: Boost router convergence with sdn. *SIGCOMM Comput. Commun. Rev.*, 45(5):341–342.

Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., and Banerjee, S. (2011). Devoflow: Scaling flow management for high-performance networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):254–265.

Feamster, N., Balakrishnan, H., Rexford, J., Shaikh, A., and van der Merwe, K. (2004). The Case for Separating Routing from Routers. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, Portland, OR.

Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S. P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E. (2014). Sdx: A software defined internet exchange. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 551–562, New York, NY, USA. ACM.

Luiz Fernando T. de Farias, Morganna C. Diniz, S. C. d. L. (2014). Uma abordagem para redução da tabela de encaminhamento sob a ótica da interface de saída dos pacotes. In *XIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance 2014)*, Porto Alegre/RS. SBC.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Nascimento, M. R., Rothenberg, C. E., Salvador, M. R., Corrêa, C. N. A., de Lucena, S. C., and Magalhães, M. F. (2011). Virtual routers as a service: The routeflow approach leveraging software-defined networks. In *Proceedings of the 6th International Conference on Future Internet Technologies*, CFI '11, pages 34–37, New York, NY, USA. ACM.

Pingping Lin, Jun Bi, H. H. (2014). Btsdn: Bgp-based transition for the existing networks to sdn. IEEE 2014, Shanghai. IEEE.

Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., Corrêa, C. N. A., Cunha de Lucena, S., and Raszuk, R. (2012). Revisiting routing control platforms with the eyes and muscles of software-defined networking. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 13–18, New York, NY, USA. ACM.

Society, T. I. (2006). Bgp route reflection: An alternative to full mesh internal bgp (ibgp). Website. `https://www.ietf.org/rfc/rfc4456.txt`.

Trilha Principal do SBRC 2016
Sessão Técnica 23
OSN e medições

Intermediação por Espalhamento: Caminhos Quase Mais Curtos Também Importam

Dianne S. V. Medeiros[1,3], Miguel Elias M. Campista[1], Nathalie Mitton[2], Marcelo Dias de Amorim[3], Guy Pujolle[3]

[1]GTA / PEE-COPPE / DEL-Poli (UFRJ) – Brasil
[2]FUN Team (INRIA Lille-Nord Europe) – França
[3]LIP6/CNRS (UPMC Sorbonne Universités) – França

{dianne, miguel}@gta.ufrj.br, nathalie.mitton@inria.fr,
{marcelo.amorim, guy.pujolle}@lip6.fr

Abstract. *Betweenness centrality metrics based on shortest paths completely neglect the contribution of nodes that are still central, but only participate in* quasi-*shortest paths. The idea of this paper is to consider such not so central nodes, assigning them more importance. To this end, this paper proposes the spread betweenness centrality (ρ-betweenness), which assigns weights to each node based on how longer is the path it belongs to compared with the shortest one. This weigth depends on the spreadness parameter ρ, that determines the maximum tolerable difference between the lengths of the shortest path and the* quasi-*shortest path. The metric analysis is formalized through the comparison between the results of the spread, traditional and distance scaled betweenness, for three real datasets and an hybrid one. Results show that the ρ-betweenness improves network connectivity and it is able to identify poorly classified nodes and to assign more importance to central nodes not necessarily in shortest paths.*

Resumo. *Métricas de centralidade de intermediação baseadas em caminhos mais curtos desprezam completamente a contribuição de nós ainda centrais, mas que estão em caminhos quase mais curtos. A ideia deste artigo é considerar esses nós não tão centrais, atribuindo-lhes maior importância. Para tanto, propõe-se a intermediação por espalhamento (ρ-intermediação), que atribui um peso a cada nó com base em quão mais longo é o caminho que pertence em relação ao mais curto. Esse peso depende do parâmetro de espalhamento ρ, que determina a diferença máxima tolerável entre os comprimentos do caminho quase mais curto e o mais curto. A análise da métrica é formalizada através da comparação entre os resultados da intermediação por espalhamento, tradicional e escalonada por distância, para três conjuntos de dados reais e um híbrido. Os resultados mostram que a ρ-intermediação melhora a conectividade da rede e é capaz de identificar nós mal classificados e atribuir maior importância a nós centrais que não necessariamente estão em caminhos mais curtos.*

1. Introdução

A representação de redes como grafos permite identificar a influência dos nós na própria rede. Métricas da teoria de grafos, permitem quantificar a importância dos

Os autores agradecem ao CNPq pelo financiamento parcial deste trabalho.

nós no sistema e, assim, aprimorar a utilização dos recursos disponíveis. Essa importância é frequentemente calculada através de métricas de centralidade [Freeman, 1978, Hui et al., 2008, Opsahl et al., 2010, Wehmuth e Ziviani, 2013], que classificam os nós de acordo com a sua posição topológica na rede. Para esse fim, é bastante comum utilizar a *centralidade de intermediação* (*betweenness centrality*), ou apenas *intermediação*, que se relaciona ao número de caminhos mais curtos dos quais um nó participa. Considera-se que quanto maior for esse número, maior é o poder centralizador do nó [Freeman, 1978].

O conceito de intermediação tem grande importância em alguns aplicações de redes de computadores, como mecanismos de eleição e protocolos de roteamento e encaminhamento, que frequentemente tomam decisões baseadas no *caminho mais curto* entre pares origem-destino. Contudo, a atribuição de importância a cada nó com base apenas no número de caminhos mais curtos, conforme a definição tradicional de intermediação, pode ocasionar uma classificação equivocada dos nós. Isso pode ocorrer ao se classificar um nó que participa de uma pequena quantidade de caminhos mais curtos como mais importante do que outro nó que participa de uma quantidade ainda menor de caminhos mais curtos, mas que está presente em inúmeros caminhos quase tão curtos.

Diversos autores [Freeman et al., 1991, Newman, 2005, Borgatti e Everett, 2006] já questionaram a utilização apenas de caminhos ideais para quantificar a importância dos nós. Este trabalho também acredita que isso limita a aplicabilidade da métrica e propõe a *centralidade de intermediação por espalhamento*, ou ρ-intermediação. Essa proposta estende a definição da centralidade de intermediação tradicional para considerar os nós que estão em caminhos não necessariamente tão curtos quanto o mais curto, mas que, ainda assim, apresentam um papel central na rede. De forma sucinta, a computação da ρ-intermediação de um nó v_k tem como base a relação entre o comprimento do caminho mais curto conectando um dado par origem-destino e o tamanho do caminho mais curto entre esses mesmos nós, passando por v_k. A ideia é atribuir um peso aos nós que estão em caminhos quase mais curtos, de forma inversamente proporcional ao comprimento do caminho mais curto que passa por ele. Entretanto, a diferença entre os comprimentos é limitada a um determinado valor ρ, o espalhamento tolerado da centralidade da rede.

A métrica proposta é avaliada através de comparações com as definições mais simples de intermediação, a tradicional [Freeman, 1977] e a escalonada por distância [Borgatti e Everett, 2006]. Verifica-se o impacto da ρ-intermediação utilizando-se três conjuntos de dados reais e um híbrido. Primeiramente, comparam-se os ranqueamentos obtidos para cada tipo de intermediação. Em seguida, investiga-se o comportamento desse ranqueamento no tempo, a fim de verificar o impacto da ρ-intermediação na estabilidade da rede. Os resultados mostram que a ρ-intermediação é capaz de identificar nós mal classificados de acordo com a visão tradicional e a escalonada por distância. A métrica proposta também se mostra útil para desempatar nós classificados na mesma posição pelas outras duas métricas, mas que de fato não apresentam a mesma importância para a rede. Ainda, a ρ-intermediação reduz a frequência de desconexões de nós em posições mais baixas e aumenta a frequência com que nós mantêm-se sempre conectados.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta a notação e as definições empregadas. A Seção 3 discute os trabalhos relacionados e revisa a definição das métricas de intermediação utilizadas. Na Seção 4 descreve-se o problema estudado enquanto a Seção 5 apresenta a proposta deste trabalho. Os conjuntos de dados são des-

critos na Seção 6 e os resultados obtidos são discutidos na Seção 7. Por fim, a Seção 8 conclui este trabalho e apresenta as direções futuras.

2. Notações e Definições

Uma rede pode ser modelada como um grafo ponderado $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, onde $\mathcal{V}$ representa o conjunto de vértices e $\mathcal{E}$ o de arestas. Cada vértice $v_i \in \mathcal{V}$ e cada aresta $\varepsilon_{i,j} \in \mathcal{E}$ representam, respectivamente, um nó da rede e um enlace entre um par de nós $[v_i, v_j]$. Cada aresta possui um peso $\omega_{i,j} \in \mathbb{R}_+$, que representa o custo do enlace. Os enlaces $\varepsilon_{i,j}$ e $\varepsilon_{j,i}$ existem simultaneamente se, e somente se, a topologia da rede for simétrica. Caso contrário, a existência de um não garante a existência do outro. A seguir, são apresentadas definições importantes para a compreensão da proposta deste trabalho.

Definição 1. Caminho: *Um caminho $p_{(.)}$ de v_1 até v_L é uma sequência de nós distintos em que qualquer par consecutivo está conectado por um enlace.*

Um caminho $p_{1,L}$ entre a origem, v_1, e o destino, v_L, possui comprimento total $\Delta L = L - 1$ enlaces, com $L \in \mathbb{N}^*$, e é dado por $p_{1,L} = \langle v_1, v_2, ..., v_{L-1}, v_L \rangle$, onde $\{v_1, v_2, ..., v_{L-1}, v_L\} \subseteq \mathcal{V}$ e $\{\varepsilon_{1,2}, ..., \varepsilon_{L-1,L}\} \subseteq \mathcal{E}$. O *caminho mais curto*, doravante *geodésica*, entre esses nós é aquele em que ΔL assume o menor valor possível, ΔL^*.

Definição 2. Caminho quase mais curto: *O caminho é "quase mais curto" se a diferença entre ΔL e ΔL^* for menor do que um limite ρ.*

Como mais de um caminho pode existir entre um mesmo par de nós, o número de geodésicas entre esses nós é representado por $n^*_{1,L}$. Para os caminhos quase mais curtos, considera-se apenas os mais curtos dentre eles, $n_{1,L}$. Observe que, qualquer modificação na sequência de nós do caminho $p_{1,L}$ origina um caminho distinto $p'_{1,L}$. Assim, $p_{1,4} = \langle v_1, v_2, v_3, v_4 \rangle$ é um caminho diferente de $p'_{1,4} = \langle v_1, v_3, v_2, v_4 \rangle$.

Definição 3. Custo total de um caminho: *O custo total de um caminho $p_{1,L}$ é uma função composta $f : \omega_{1,2}, ..., \omega_{L-1,L} \rightarrow \mathbb{R}_+$.*

O caminho que apresenta o menor custo total é o *melhor* caminho. Este trabalho utiliza o número de saltos como métrica de custo, de forma que o melhor caminho é a geodésica, representada por $p^*_{1,L}$. Como as comunicações nesse modelo de rede ocorrem salto a salto, dada a geodésica $p^*_{1,L} = p_{1,L}$, o nó v_1 precisa conhecer apenas o seu vizinho imediato, v_2, que faz parte do menor caminho até v_L. O custo dessa geodésica é representado por $\delta^*_{1,L}$, enquanto o custo de um caminho quase mais curto entre os mesmos nós é dado por $\delta_{1,L}$.

Definição 4. Espalhamento: *O espalhamento ρ é a diferença máxima tolerável entre os comprimentos ΔL e ΔL^*.*

O espalhamento ρ provê um ajuste fino à ρ-intermediação, para considerar em seu cálculo apenas os menores caminhos quase mais curtos cujo comprimento máximo não ultrapasse $\rho - \Delta L^*$. A ideia é que, dada uma geodésica $p^*_{1,L}$, é pouco razoável considerar a existência de volume significativo de informação trafegando por caminhos quase mais curtos, em que $\Delta L \gg \Delta L^*$. A Tabela 1 sumariza as informações dessa seção.

Tabela 1. Notação utilizada neste trabalho.

Parâm.	Descrição	Parâm.	Descrição
v_i	Nó qualquer da rede	$\varepsilon_{i,j}$	Enlace entre os nós v_i e v_j
$\omega_{i,j}$	Peso do enlace entre os nós v_i e v_j	$p_{i,j}$	Caminho entre os nós v_i e v_j
$p^*_{i,j}$	Geodésica entre os nós v_i e v_j	ΔL	Comprimento de um caminho entre os nós v_i e v_j
ΔL^*	Comprimento da geodésica entre os nós v_i e v_j	$n_{i,j}$	Número de caminhos quase mais curtos entre os nós v_i e v_j
$n^*_{i,j}$	Número de geodésicas entre os nós v_i e v_j	$\delta_{i,j}$	Custo do caminho quase mais curto entre os nós v_i e v_j, em número de enlaces
$\delta^*_{i,j}$	Custo da geodésica entre os nós v_i e v_j, em número de enlaces	ρ	Epalhamento, diferença máxima aceitável entre L e L^*

3. Revisitando a Centralidade de Intermediação

A centralidade de intermediação expressa a influência que um dado nó poderia exercer sobre os outros nós da rede. Ela se baseia na computação do número de geodésicas que passam por um nó, considerando todos os possíveis pares de nós existentes. Contudo, contabilizar apenas as geodésicas pode limitar a métrica [Freeman et al., 1991, Newman, 2005, Borgatti e Everett, 2006, Opsahl et al., 2010]. Por exemplo, em um grafo ponderado, no qual se representa a intensidade da ligação entre nós adjacentes, nem sempre o caminho mais curto será aquele com menor custo. Isso dificulta o emprego da intermediação tradicional para quantificar a importância dos nós, uma vez que ela não considera a intensidade da relação entre os nós. Freeman et al. [Freeman et al., 1991] e Opsahl et al. [Opsahl et al., 2010] atacam essa deficiência da métrica tradicional, possibilitando quantificar a importância dos nós em grafos ponderados nos quais nem sempre o nó com maior intermediação será aquele que se encontra no maior número de geodésicas.

Outra abordagem é utilizada por Newman [Newman, 2005] para tratar do mesmo problema. Apesar de também considerarem que em alguns casos a informação trafega por outros caminhos que não são os mais curtos, os autores defendem que isso ocorre porque nessas redes a informação não conhece o caminho ideal ou não tem um destino específico, e não devido à diferença na intensidade das relações entre os nós. Os autores propõem, então, a intermediação por caminhos aleatórios, na qual contabilizam tanto os caminhos mais curtos quanto os não ideais. Outro ponto de vista proposto por Borgatti et al. [Borgatti e Everett, 2006] é que ao contabilizar somente os caminhos mais curtos, não se deve atribuir o mesmo peso a todos eles, uma vez que podem apresentar comprimentos diferentes e a informação tende a se concentrar em caminhos menos longos. Assim, os autores propõem a intermediação escalonada por distância, na qual a contribuição de cada caminho deve ser ponderada pelo seu comprimento de forma inversamente proporcional.

Este trabalho também defende a ideia principal dos trabalhos apresentados de que nem sempre a informação trafega pelos caminhos ideais. No entanto, nossa abordagem difere ao propor uma expressão fechada para a intermediação ponderada, considerando não apenas os caminhos mais curtos, mas também aqueles que diferem de um pequeno valor no comprimento quando comparados ao ideal, sem no entanto ter que contabilizar todos os caminhos existentes. Nas subseções seguintes, são revisadas a definição formal da intermediação tradicional e da escalonada por distância, selecionadas para comparação

com a nossa proposta por serem as métricas mais simples dentre as apresentadas.

3.1. Centralidade de intermediação tradicional

Tradicionalmente a intermediação é definida para grafos não ponderados, podendo ser empregada em grafos simétricos ou assimétricos. Sua definição formal é dada por:

$$b(v_k) = \sum_{\substack{i=1 \\ i \neq k}}^{|\mathcal{V}|} \sum_{\substack{j=1 \\ j \neq i \\ j \neq k}}^{|\mathcal{V}|} \frac{n^*_{i,j}(v_k)}{n^*_{i,j}}, \tag{1}$$

onde $n^*_{i,j}(v_k)$ é o número de geodésicas entre v_i e v_j que possuem v_k como nó intermediário. Note que para ser considerado uma geodésica, um caminho $p_{1,L}$ deve apresentar obrigatoriamente um custo total igual a $\delta^*_{1,L}$. Assim, qualquer outro caminho cujo custo total seja maior do que $\delta^*_{1,L}$ será ignorado nessa computação. Consequentemente, a seguinte restrição pode ser estabelecida:

$$\begin{cases} \frac{n^*_{i,j}(v_k)}{n^*_{i,j}} > 0, & \text{se } \exists p_{i,j}(v_k) \mid \delta_{i,j}(v_k) = \delta^*_{i,j}, \\ \frac{n^*_{i,j}(v_k)}{n^*_{i,j}} = 0, & \text{caso contrário.} \end{cases} \tag{2}$$

Note que $n^*_{i,j}(v_k) = 0$, já que não há nenhum caminho $p_{i,j}(v_k)$ tal que $\delta_{i,j}(v_k) = \delta^*_{i,j}$.

3.2. Centralidade de intermediação escalonada por distância

A ideia da métrica é tornar a contribuição de cada caminho inversamente proporcional ao seu comprimento, atribuindo importância distinta para cada caminho considerado na Equação 1. Sua definição formal é dada pela Equação 3.

$$b_{\lambda_{i,j}}(v_k) = \sum_{\substack{i=1 \\ i \neq k}}^{|\mathcal{V}|} \sum_{\substack{j=1 \\ j \neq i \\ j \neq k}}^{|\mathcal{V}|} \frac{1}{\Delta L} \times \frac{n^*_{i,j}(v_k)}{n^*_{i,j}}. \tag{3}$$

4. Caracterização do Problema

A centralidade de intermediação, em sua visão tradicional, muitas vezes deixa de considerar nós aparentemente mais importantes para a conectividade da rede, em detrimento de outros. Isso ocorre porque tradicionalmente a métrica utiliza apenas os caminhos mais curtos para definir a importância dos nós. Esse problema está ilustrado na Figura 1, na qual v_c é o principal responsável por manter os componentes A e B conectados. Em caso de falha desse nó, v_p deveria ser o mais importante, uma vez que ele seria então o responsável por manter a conectividade. No entanto, na visão tradicional da intermediação, a centralidade de v_b é maior do que a de v_p porque conecta v_f a todos os outros nós. Apesar disso, v_b não consegue manter a conectividade de um componente maior. A Tabela 2 compara as intermediações computadas para v_c, v_b e v_p, usando as intermediações tradicional, escalonada por distância e por espalhamento, considerando-se apenas 5 nós totalmente conectados nas redes azul e verde representadas na Figura 1. A

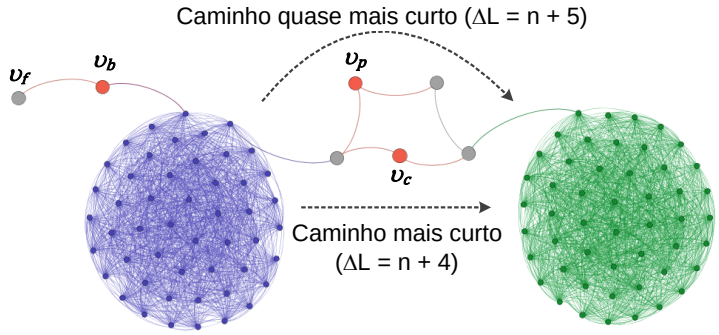


Figura 1. Exemplo de rede na qual a centralidade de intermediação tradicional elege como mais importante o nó v_b, aparentemente menos significativo para manter a conectividade de um componente maior.

ρ-intermediação captura melhor a centralidade do nó, mesmo que ele não pertença tantas vezes às geodésicas entre pares de nós na rede. Esse resultado também poderia ser utilizado, por exemplo, para determinar quais nós seriam capazes de formar bons caminhos para realizar balanceamento de carga na rede.

A Figura 2 ilustra o grafo de Freeman [Freeman e Freeman, 1979], onde os nós posicionados mais centralmente possuem maior intermediação. Existe um grande número de caminhos entre dois nós na rede, mas eles podem ser subutilizados ao se considerar apenas as geodésicas. Percebe-se a quantidade de caminhos desprezados devido à definição tradicional da intermediação ao se computar a diferença de tamanho entre a geodésica $p^*_{i,j}$, de custo $\delta^*_{i,j}$, e o caminho mais curto intermediado por v_k, $p_{i,j}(v_k)$, de custo $\delta^*_{i,j}(v_k)$ resultante da soma entre $\delta^*_{i,k}$ e $\delta^*_{k,j}$. Se $(\delta^*_{i,k} + \delta^*_{k,j}) - \delta^*_{i,j} = 0$, então v_k pertence a uma geodésica. Essa diferença é referenciada por $\Delta = (\delta^*_{i,k} + \delta^*_{k,j}) - \delta^*_{i,j}$, sendo ρ o limite superior de Δ para considerar o caminho na computação da métrica proposta.

A Figura 3 mostra a distribuição de Δ no grafo de Freeman. O eixo x e o eixo y representam, respectivamente, o nó intermediário e todos os possíveis valores de Δ para esse nó. Logo, as linhas verticais ilustram a distribuição de Δ para cada nó, considerando todos os pares origem-destino na rede. A escala de cores indica a frequência relativa de ocorrência para um dado Δ, de forma que quanto mais escura a cor, maior é a frequência. Observa-se claramente na Figura 3 que frequentemente existem valores $\Delta > 0$. Além disso, menos de 20% ($0 < \text{ID} \leq 5$) dos nós encontram-se com frequência em caminhos mais curtos. Conclui-se, então, que existe um número considerável de nós que podem nunca intermediar uma comunicação devido à definição tradicional da intermediação.

Na Figura 4, observa-se a frequência com que uma dada distância é encontrada no grafo de Freeman, considerando-se todos os nós da rede. De forma geral, nota-se que $\Delta = 1$ e $\Delta = 2$ ocorrem com frequência muito maior do que $\Delta = 0$, enquanto o valor

Tabela 2. Comparação entre as três métricas para os nós na rede da Figura 1.

Nó	Tradicional	Escalonada	Espalhamento
v_c	48	9,4	53,1
v_p	8	2,2	67,8
v_b	15	3,5	15,0

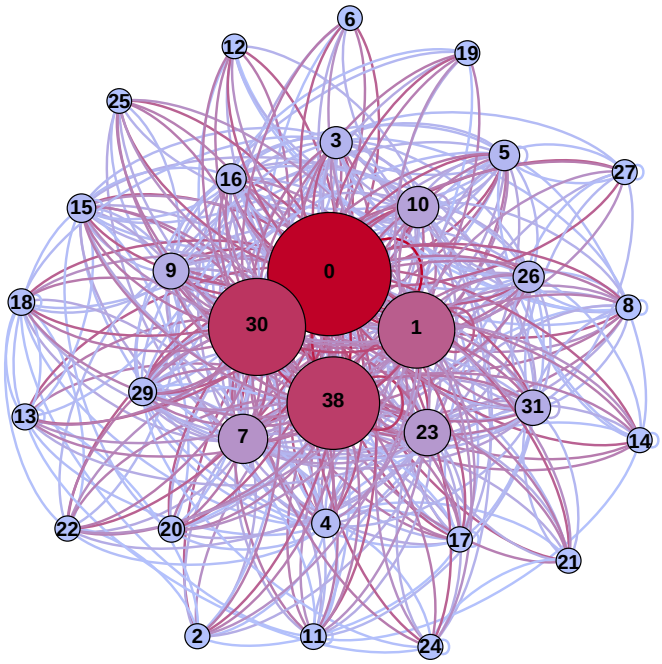


Figura 2. Grafo de Freeman. Nós em posições mais centrais possuem maior valor de intermediação tradicional.

máximo $\Delta = 4$ é quase desprezível. Esses resultados indicam que nós em caminhos alternativos devem ser considerados no cálculo da intermediação, mas apenas os que estão em caminhos não muito mais longos do que a geodésica. Em algumas topologias o problema se agrava ao se aumentar o número de pares origem-destino e a quantidade de caminhos entre eles, que pode levar a uma concentração de tráfego em uma fração muito pequena de nós se os nós intermediários forem escolhidos utilizando-se a intermediação tradicional.

5. Centralidade de Intermediação por Espalhamento

O objetivo desse trabalho é capturar o potencial de nós intermediários, mesmo quando não pertencem a uma geodésica, revelando assim nós na rede ignorados pela visão tradicional da intermediação. Para tal, este trabalho propõe uma variante da intermediação tradicional, a *centralidade de intermediação por espalhamento*, que considera, além das geodésicas, caminhos quase mais curtos entre qualquer par de nós na rede. Além disso, a contribuição de cada caminho para a métrica é inversamente proporcional ao tamanho do caminho. Dessa forma, um nó intermediário que está em um caminho *pouco mais longo* do que a geodésica terá mais importância do que se estivesse em um caminho *muito maior* do que ela. A visão tradicional da intermediação não permite essa diferenciação.

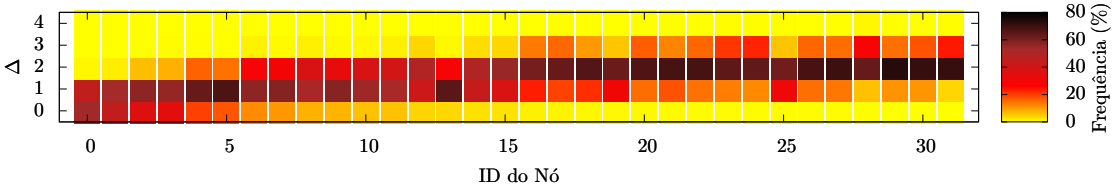
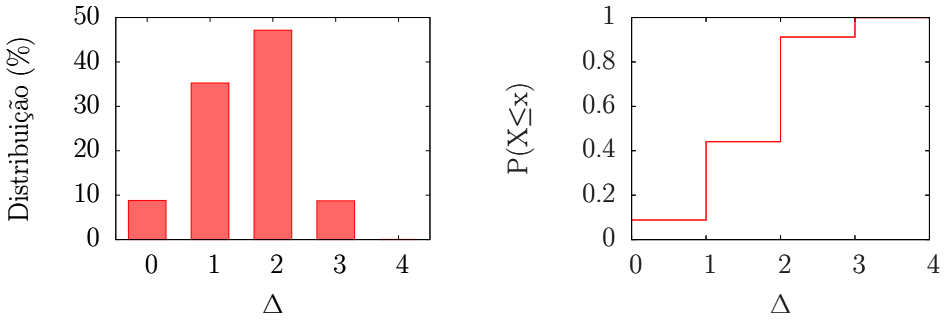


Figura 3. Distribuição da diferença entre os comprimentos da geodésica e dos caminhos que passam por qualquer nó intermediário no grafo de Freeman.

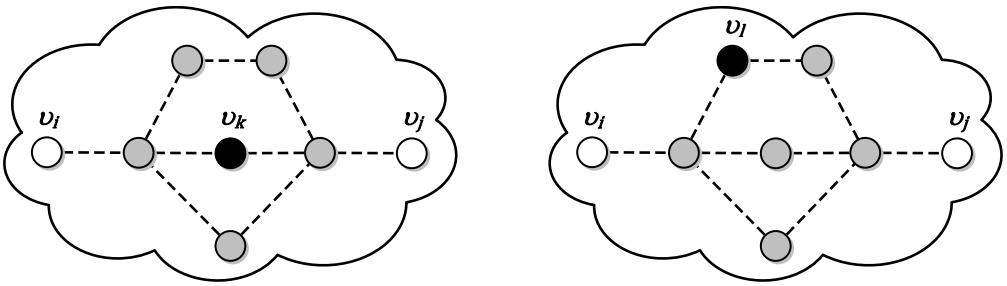


(a) Distribuição relativa de Δ. (b) Distribuição cumulativa de Δ.

Figura 4. Distribuição do parâmetro Δ para o grafo de Freeman.

A ρ-intermediação não considera a razão $n^*_{i,j}(v_k)/n^*_{i,j}$, que se baseia no *número* de geodésicas. Em vez disso, ela considera a razão entre o custo da geodésica entre v_i e v_j e o custo do caminho quase mais curto entre esse mesmo par que passa por um nó intermediário v_k. Um caminho quase mais curto, $p_{i,j}(v_k)$, é composto pela geodésica de v_i a v_k e de v_k a v_j. Assim, a relação entre os custos supracitados é definida como $\delta^*_{i,j}/(\delta^*_{i,k} + \delta^*_{k,j})$. Se a diferença entre ΔL e ΔL^* for maior do que o espalhamento ρ, o caminho quase mais curto será ignorado. Note que utilizar o número de saltos como custo não limita a métrica, que se permite generalizar para qualquer tipo de peso.

A Figura 5 ilustra a diferença entre a abordagem tradicional, escalonada por distância e a proposta deste trabalho, utilizando o número de saltos como métrica para o custo do caminho. Na Figura 5(a), o nó v_k se encontra em uma das duas geodésicas entre v_i e v_j. Logo, para a intermediação tradicional e escalonada por distância tem-se que $n^*_{i,j}(v_k)/n^*_{i,j} = 1/2$, sendo que para a última métrica ainda acrescenta-se o peso $1/\Delta L^* = 4$. Na Figura 5(b), v_l está em um caminho alternativo entre os mesmos nós, que não é uma geodésica para esse par de nós. Enquanto ambas a visão tradicional e a escalonada por distância ignorariam o potencial desse nó para o par analisado, a ρ-intermediação consideraria a relação $\delta_{i,j}/(\delta_{i,l} + \delta_{l,j}) = 4/5$. Para também levar em conta a possibilidade de existir mais de um caminho com mesmo comprimento passando por v_l, a relação entre os custos é ainda escalonada pelo número de caminhos de comprimentos iguais. Assim,



(a) Intermediação tradicional ou escalonada por distância. (b) Intermediação por espalhamento.

Figura 5. A intermediação tradicional e a escalonada por distância se baseiam no número de caminhos, enquanto a ρ-intermediação utiliza os custos desses caminhos. A diferença entre as duas abordagens é evidenciada quando o nó intermediário se encontra em um caminho quase mais curto.

define-se formalmente a ρ-intermediação como:

$$\rho(v_k) = \sum_{\substack{i=1 \\ i \neq k}}^{|\mathcal{V}|} \sum_{\substack{j=1 \\ j \neq i \\ j \neq k}}^{|\mathcal{V}|} \frac{n^*_{i,j}}{n_{i,j}(\delta^*_{i,k,j})} \times \frac{\delta^*_{i,j}}{\delta^*_{i,k} + \delta^*_{k,j}}, \tag{4}$$

onde $n^*_{i,j}$ é o número de geodésicas entre v_i e v_j e $n_{i,j}(\delta^*_{i,k,j})$ é o número de caminhos de mesmo comprimento que o menor caminho quase mais curto $p_{i,j}(v_k)$. A ρ-intermediação como definida na Equação 4 guarda as propriedades a seguir:

- Seu valor aumenta com a centralidade do nó;
- Considera o número de múltiplos caminhos de mesmo comprimento daquele que passa pelo nó intermediário;
- É computada apenas para os nós que estão no mesmo componente.

Tanto o limite inferior quanto o superior para a métrica proposta dependem das relações entre o número de caminhos, $n^*_{i,j}/n_{i,j}(\delta^*_{i,k,j})$, e entre os custos dos caminhos, $\delta^*_{i,j}/(\delta_{i,k} + \delta_{k,j})$. O limite inferior depende ainda do espalhamento. No pior caso para o limite inferior, se $n_{i,j}(\delta^*_{i,k,j}$ for muito maior do que $n^*_{i,j}$, ou se o custo do caminho por v_k for muito maior do que o da geodésica entre v_i e v_j, ou ainda, se $(\delta^*_{i,k} + \delta^*_{k,j}) - \delta^*_{i,j} > \rho$, o termo correspondente tenderá a ser nulo. Já o limite superior poderá ser igual a 1 quando todos os caminhos quase mais curtos forem, na verdade, geodésicas, ou igual a $n^*_{i,j} \cdot \delta^*_{i,j}/(\delta_{i,k} + \delta_{k,j})$, se existir apenas um caminho quase mais curto, ou ainda depender de ambas as relações em um caso mais geral.

Como a ρ-intermediação considera também caminhos um pouco mais longos do que a geodésica, contabilizar todos os menores caminhos quase mais curtos entre um par de nós intermediados ou não por v_k pode se tornar uma tarefa inviável devido à complexidade. Assim, utiliza-se o espalhamento ρ para limitar o algoritmo de busca em profundidade (Depth-First Search - DFS) utilizado neste trabalho, reduzindo sua complexidade. Ainda assim, é necessário computar todos os caminhos de comprimento $\delta_{i,k} + \delta_{k,j}$ entre cada par $[v_i, v_j]$, o que pode ser uma tarefa custosa para grafos com muitos enlaces. Logo, além de diferenciar se um nó intermediário pertence ou não a uma geodésica entre v_i e v_j como a intermediação tradicional, a ρ-intermediação faz a mesma diferenciação para os *menores caminhos quase mais curtos* de comprimento $\delta_{i,k} + \delta_{k,j}$ possíveis entre esse mesmo par. A importância de cada termo relacionado ao nó intermediário é reduzida em função da relação entre o número de geodésicas e o número de menores caminhos quase mais curtos com o mesmo comprimento de $p_{i,j}(v_k)$ existentes.

Por fim, assim como a intermediação tradicional, a ρ-intermediação também requer o conhecimento do custo dos caminhos, que podem ser infinitos em componentes desconectados. Dessa forma, a métrica proposta é computada apenas para nós que estão no mesmo componente para evitar problemas relacionados à distância entre nós que estão em componentes desconectados.

6. Conjunto de Dados

A ρ-intermediação é avaliada através da utilização de conjuntos de dados com características distintas, a fim de manter a generalidade da métrica. Os conjuntos de dados selecionados estão descritos adiante e suas características estão sumarizadas na Tabela 3.

- **Freeman's EIES**: apresenta o relacionamento em um grupo de acadêmicos [Freeman e Freeman, 1979]. Os dados são compostos de traços relacionados a mensagens eletrônicas enviadas através do Sistema Eletrônico de Troca de Informação (EIES). Uma aresta entre dois nós v_i e v_j existe apenas se v_i tiver enviado uma mensagem para v_j durante o período do experimento.
- **Dolphins**: fornece a relação de associação entre golfinhos de Doubtful Sound, Nova Zelândia [Lusseau et al., 2003]. Cada nó na rede corresponde a um golfinho e a interação entre eles é representada por uma aresta não direcional $\varepsilon_{i,j}$.
- **PhD Students**: é uma rede de relações entre doutorandos e seus orientadores [Johnson, 1984]. Uma aresta existe de v_i para v_j se v_i for orientador de v_j.
- **TAPASCologne Dataset**: modela o tráfego de veículos na cidade de Colônia [Uppoor e Fiore, 2011]. Neste trabalho, utiliza-se 10 instantâneos do subconjunto disponibilizado pelo projeto, sendo que cada um deles possui uma quantidade distinta de nós, arestas e componentes. Cada nó representa um veículo e uma aresta existe entre dois veículos somente se eles estiverem geograficamente a menos de 50 metros de distância um do outro. Nós desconectados foram desconsiderados, uma vez que sua intermediação é nula em qualquer abordagem.

7. Resultados

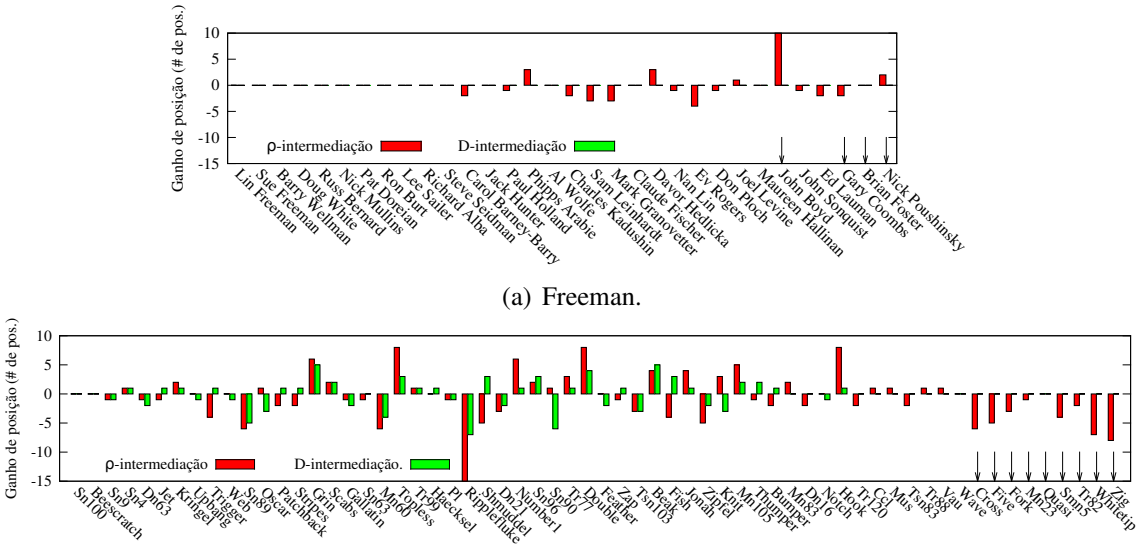
Esta seção apresenta o impacto da ρ-intermediação nos conjunto de dados selecionados. Utiliza-se $\rho = 3$ para o espalhamento, porque na Seção 4 verificou-se para o grafo de Freeman que valores $\Delta \geq 4$ são pouco frequentes. Assim, considera-se apenas os menores caminhos mais curtos que obedeçam à relação $\Delta L \leq \Delta L^* + 3$. O objetivo inicial é analisar o comportamento do ranqueamento dos nós obtido em cada métrica. Em seguida, verifica-se o impacto da métrica proposta na rede. Note que nos resultados obtidos considera-se que nós com a mesma intermediação estão empatados na mesma posição.

7.1. Impacto da ρ-intermediação na classificação dos nós

A variação no ranqueamento dos nós é investigada tanto para a ρ-intermediação quanto para a intermediação escalonada por distância (D-intermediação). Para tal, verifica-se o ganho de posição do nó para cada métrica em relação à intermediação tradicional. Um ganho negativo implica perda de importância, enquanto um ganho positivo indica a ascensão do nó. A Figura 6 mostra que ambas as métricas são capazes de modificar a classificação dos nós. Na Figura 6(a), o eixo-X está organizado em ordem crescente de classificação segundo a visão tradicional para o conjunto de dados de Freeman, sendo "Lin Freeman" o nó mais central. Observa-se que a ρ-intermediação modifica a classificação de alguns nós, especialmente aqueles que estão em posições mais baixas, indicando que a intermediação tradicional pode subestimar a centralidade de nós que não estão ou que participam de poucas geodésicas. Por exemplo, "John Boyd" é reclassificado 10 posições

Tabela 3. Sumário das propriedades dos conjuntos de dados.

Nome	# de vért.	# de arestas	# de componentes	Diâm.	Simetria
Freeman's EIES	32	460	Fraco=1,Forte=1	3	Assim.
Dolphins	62	159	1	8	Sim.
PhD. Students	1.025	1.043	Fraco=1, Forte=1.025	10	Assim.
TAPASCologne	1.584–1.916	1.573–2.044	550–640	7–15	Sim.



(a) Freeman.

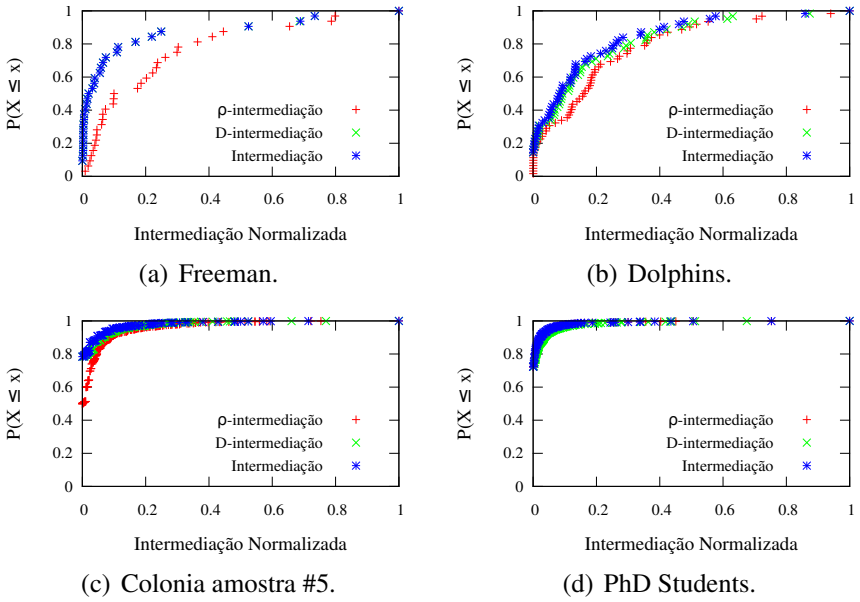
(b) Dolphins.

Figura 6. Ganho de posição dos nós em relação ao ranqueamento da intermediação tradicional, quando comparado ao da D- e da ρ-intermediação.

acima, segundo a ρ-intermediação. Adicionalmente, segundo a intermediação tradicional, "Gary Coombs", "Brian Foster" e "Nick Poushinsky" estão empatados na posição 29, com intermediação 0. Ao contrário da D-intermediação, que mantém esses nós empatados, com intermediação 0, a ρ-intermediação é capaz de lhes atribuir novas posições. A reclassificação dos nós também ocorre para o conjunto de dados Dolphins, conforme ilustra a Figura 6(b). Diversos nós são rebaixados e promovidos utilizando a D- e a ρ-intermediação. Novamente observa-se que um número significativo de nós classificados em posições mais baixas são reclassificados utilizando a métrica proposta, enquanto a D-intermediação parcamente modifica o ranqueamento. Além disso, nós antes empatados com intermediação nula (nó "Cross" em diante) são redistribuídos no ranqueamento.

A Figura 7 mostra a função de distribuição cumulativa dos valores das intermediações para as três métricas. Para cada métrica, o valor da intermediação foi normalizado segundo o valor máximo encontrado para o conjunto de dados. Assim, as três métricas podem ser plotadas usando uma mesma faixa de valores para o eixo-X. No conjunto de dados de Freeman, Figura 7(a), as curvas para a intermediação tradicional e a D-intermediação são coincidentes, corroborando o resultado ilustrado na Figura 6(a), em que o ganho de posição de cada nó classificado pela D-intermediação é nulo em relação ao ranqueamento da intermediação tradicional. Ou seja, ambas as métricas apresentam o mesmo ranqueamento. Além disso, observa-se que aproximadamente 40% dos nós apresentam pouca ou nenhuma importância para a rede. Enquanto isso, a ρ-intermediação atribui pouca ou nenhuma importância para menos de 5% dos nós. Assim como na Figura 7(a), as Figuras 7(b) e 7(c) mostram que a ρ-intermediação reduz a porcentagem de nós considerados pouco ou não importantes para a rede, quando comparada à intermediação tradicional e à D-intermediação.

A Figura 7(d) mostra um comportamento diferente para a distribuição cumulativa dos valores de intermediação para o grafo PhD Students. As curvas para as três métricas são praticamente coincidentes. Isso ocorre porque o conjunto de dados PhD Students é um



(a) Freeman. (b) Dolphins.

(c) Colonia amostra #5. (d) PhD Students.

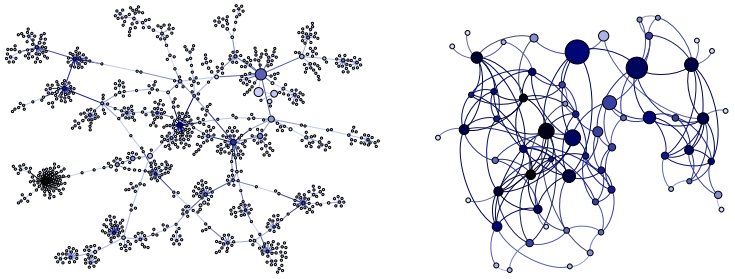
Figura 7. Distribuição cumulativa do valor normalizado da intermediação computado para os quatro conjuntos de dados utilizando ρ-intermediação, D-intermediação e intermediação tradicional.

grafo dirigido que apresenta um grande número de vértices que nunca, ou quase nunca, intermedeiam comunicações. O número de caminhos possíveis entre dois nós na rede é pequeno, porque as arestas apontam de um orientador para um estudante e poucos nós representam os dois papéis. Esse comportamento pode ser observado na Figura 8, que compara a organização dos nós nos conjuntos de dados PhD Students e Dolphins. Nessa figura nós maiores possuem maior intermediação tradicional e nós mais escuros apresentam maior grau. O conjunto de dados PhD Students apresenta um grande número de nós "folha" que possuem intermediação nula porque não intermedeiam nenhuma comunicação e, portanto, qualquer que seja a métrica utilizada o valor da intermediação continuará sendo nulo. Isso não é observado no conjunto Dolphins, muito mais bem conectado e repleto de múltiplos caminhos entre dois nós quaisquer.

7.2. Impacto da ρ-intermediação na estabilidade da rede

Os resultados discutidos mostram que a ρ-intermediação consegue reclassificar nós que aparentemente deveriam ter maior importância para a conectividade da rede. A ideia principal da métrica é reduzir o número de desconexões em redes de topologia dinâmica. Nesse sentido, as 10 amostras da rede formada pelo conjunto de dados TAPAS-Cologne, tomadas a cada 10 segundos, foram utilizadas para investigar o comportamento das desconexões para as três métricas. Considera-se que uma desconexão ocorre quando um nó deixa de intermediar caminhos, isto é, sua intermediação é reduzida para 0.

A Figura 9(a) mostra a frequência com que as desconexões ocorrem nas amostras da rede TAPASCologne. O eixo-X indica o número de desconexões, enquanto o eixo-Y mostra a frequência com que aquele número de desconexões ocorreu no período estudado. Note que o valor $x = 0$ indica que não houve desconexão durante aquele período. Nesse caso, a frequência com que os nós nunca se desconectaram é muito maior para a ρ-intermediação em comparação com as outras métricas. A partir daí as desconexões são



(a) PhD Students. (b) Dolphins.

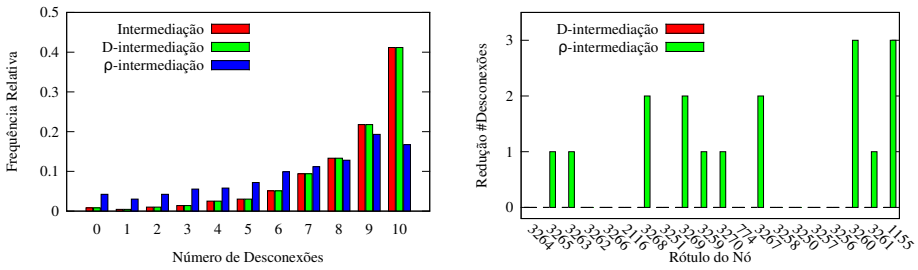
Figura 8. O grafo PhD Students forma ilhas quase isoladas com múltiplos nós "folha" em torno de nós centrais. Já o grafo Dolphins constitui uma rede bem conectada, com múltiplos caminhos possíveis entre os nós.

mais frequentes para a ρ-intermediação. No entanto, a taxa de crescimento da frequência de desconexão da ρ-intermediação é muito menor do que para a intermediação tradicional e a D-intermediação. Assim, a frequência de desconexão para essas métricas passa a ser maior a partir de 8 desconexões. A frequência torna-se muito maior para 10 desconexões, que significa, na verdade, que nas 10 amostras tomadas o nó nunca esteve conectado.

A fim de verificar se a redução na frequência de desconexões beneficia também os nós mais importantes da rede, analisa-se quantas desconexões a mais ou a menos ocorrem para os nós que inicialmente eram os top 20 da rede TAPASCologne. Na Figura 9(b) o eixo-X representa o rótulo do nó, em ordem crescente de importância. O eixo-Y indica a diferença entre o número de desconexões para ρ-intermediação e D-intermediação em comparação à intermediação tradicional. Observa-se que para os top 20 nós a D-intermediação não evita nenhuma desconexão em comparação à intermediação tradicional. Em contrapartida, a ρ-intermediação é capaz de evitar até 3 desconexões para metade dos nós que eram inicialmente os mais importantes.

8. Conclusões

Este trabalho propõe a ρ-intermediação, uma variação da centralidade de intermediação tradicional. O objetivo é atribuir importância para nós que se encontram em caminhos *quase* mais curtos, uma vez que eles podem ser fundamentais para manter a conectividade da rede, mas são completamente desconsiderados pela intermediação tradi-



(a) Frequência de desconexões. (b) Redução no número de desconexões.

Figura 9. Frequência de desconexões para as três métricas e redução no número de desconexões para os top 20 nós ao se utilizar a ρ-intermediação. A métrica proposta reduz as desconexões na rede.

cional. Para analisar o impacto da métrica foram utilizados quatro conjuntos de dados com características distintas, para os quais foram também computadas a intermediação tradicional e a escalonada por distância. Observa-se que enquanto a intermediação tradicional e a escalonada por distância induzem a pensar que nós empatados no ranqueamento possuem a mesma importância para a rede, a ρ-intermediação mostra que isso geralmente não é verdade, pois esses nós podem participar de um número diferente de caminhos quase mais curtos. Além disso, a ρ-intermediação é capaz de reduzir a frequência de desconexões, principalmente para nós classificados em posições mais baixas quando se utiliza a intermediação tradicional ou a escalonada por distância. A métrica proposta também é capaz de aumentar a frequência com que nós mantêm-se sempre conectados. Assim, a ρ-intermediação consegue reclassificar nós que aparentemente são mais importantes para a conectividade da rede e ainda mantém a conectividade tanto de nós mal classificados como daqueles que estão em posições mais importantes. Como trabalhos futuros, pretende-se verificar a influência da variação do parâmetro ρ no impacto da métrica proposta, analisar sua relação com o grau dos nós e estendê-la para redes ponderadas.

Referências

Borgatti, S. P. e Everett, M. G. (2006). A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466 – 484.

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239.

Freeman, L. C., Borgatti, S. P. e White, D. R. (1991). Centrality in valued graphs: A measure of betweenness based on network flow. *Social Networks*, 13(2):141–154.

Freeman, S. e Freeman, L. (1979). *The Networkers Network: A Study of the Impact of a New Communications Medium on Sociometric Structure*. Social sciences research reports. School of Social Sciences, University of California.

Hui, P., Crowcroft, J. e Yoneki, E. (2008). Bubble rap: Social-based forwarding in delay tolerant networks. Em *ACM Mobihoc*, p. 241–250.

Johnson, D. S. (1984). The genealogy of theoretical computer science. *SIGACT News*, 16(2):36–44.

Lusseau, D., Schneider, K., Boisseau, O., Haase, P., Slooten, E. e Dawson, S. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405.

Newman, M. J. (2005). A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39 – 54.

Opsahl, T., Agneessens, F. e Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251.

Uppoor, S. e Fiore, M. (2011). Large-scale urban vehicular mobility for networking research. Em *IEEE VNC 2011*, p. 62–69.

Wehmuth, K. e Ziviani, A. (2013). DACCER: Distributed assessment of the closeness centrality ranking in complex networks. *Computer Networks*, 57(13):2536–2548.

Empregando Entropia na Detecção de Comportamento Automatizado nos *Trend Topics* do Brasil

Adeilson Souza[1], **Eduardo Feitosa**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. Gen. Rodrigo Octávio, 6200 – 69.000-077 – Manaus – AM – Brasil

{adeilson, efeitosa}@icomp.ufam.edu.br

Abstract. *Twitter enables the creation and propagation of automatized tweets for commercial purposes, but this possibility has been used by social bots - automated accounts - for malicious purposes, especially in Trend Topics (TT). Studying the characteristics and behaviors of social bots in Twitter, this article proposes six new features, based on entropy, with high discrimination power and able to distinguish automated behavior in Brazilian TT. Using a real database containing 2.853.822 accounts and 11.294.861 tweets, we identified and extracted features of tweets (texts and user behaviors) on TT. Next, applying machine learning algorithms for supervised classification, we are capable to detect 92% of automated accounts in the used database and thus get an insight into the behavior of these users.*

Resumo. *O Twitter permite a criação e propagação de postagens automatizadas para fins comerciais, mas tal característica vem sendo utilizada indevidamente por bots sociais - contas automatizadas - para fins maliciosos, especialmente nos Tópicos de Tendência (TT). Estudando as características e comportamento dos bots no Twitter, este artigo propõe seis novas características, baseadas no conceito de Entropia, com alto poder discriminativo e capazes de distiguir o comportamento automatizado nos TT no Brasil. A partir de uma base de dados real contendo 2.853.822 contas e 11.294.861 tweets, foram identificadas e extraídas características do texto das mensagens e do comportamento dos usuários nos TT. Aplicando algoritmos de aprendizagem de máquina supervisionada para classificação, foi possível detectar 92% das contas automatizadas na base de dados utilizada e, assim, obter uma visão do comportamento desses usuários.*

1. Introdução e Motivação

A presença de perfis "não humanos" em redes socais, criados com a intenção de propagar automaticamente conteúdo não desejado vem aumentando nos últimos anos. O Twitter, por exemplo, afirmou que apenas 5% dos seus 215 milhões de usuários ativos estavam ligados a atividades automatizadas [Gara 2013], mas autores como Chu et al. [Chu et al. 2012] estimam que 50% das contas no Twitter estão ou são associadas a *bots* sociais[1].

[1]*Bots* sociais são programas automatizados que utilizam contas em redes sociais, para se passar por usuários legítimos, com o intuito de enganar ou influenciar outros usuários

De acordo com as regras do Twitter [Twitter 2014], *bots* sociais são permitidos para compartilhar conteúdo útil (notícias e eventos) e propagandas autorizadas. Contudo, vários estudos mostram que os *bots* sociais no Twitter vem sendo utilizados como canal de controle para *botnets* [Nazario 2009, Kartaltepe et al. 2010], para *phishing* [Malware Bytes 2014, Symantec 2013], para propagação de spam contendo propagandas, pornografia e código malicioso [Neal Ungerleider 2015] e também para favorecer candidatos políticos [Orcutt 2012], mesmo que existam regras e práticas para criação e uso de contas automatizadas (por exemplo, postar *tweets* automatizados repetidamente para os Tópicos de Tendência é proibido).

O Twitter fornece métodos para que usuários denunciem comportamento suspeito ou conteúdos ofensivos. Tais denúncias são investigadas e caso algo seja comprovado, as contas são suspensas e/ou o acesso pelo endereço IP referente à conta é temporariamente bloqueado. Entretanto, esse mecanismo não tem se mostrado muito eficiente, uma vez que basta os usuários maliciosos criarem uma conta diferente para continuarem enviando mensagens. No caso de denúncias envolvendo *tweets* nos TT, o resultado é ainda pior, uma vez que o processo de suspensão é lento e os TT são efêmeros e, em muitos casos, duram apenas algumas horas [Martinez-Romo and Araujo 2013]. Na literatura existem diferentes trabalhos de detecção de *bots* sociais no Twitter [Benevenuto et al. 2010, Lee et al. 2010, Stringhini et al. 2010, Wang 2010, Wang 2012] que visam minimizar e combater o uso indevido de contas automatizadas através do estudo de seu comportamento, identificando características e padrões de atividades que possam ser utilizados para desenvolver contramedidas.

A fim de contribuir com a detecção de *bots* sociais no Twitter, este artigo propõe e avalia novas características baseadas no conceito de entropia, capazes de detectar comportamento automatizado nos Tópicos de Tendência do Twitter no Brasil, através da medição dos padrões de escrita dos *tweets* postados pelos usuários. Para alcançar esse objetivo, uma base com *tweets* e dados das postagens e contas de usuários que comentam sobre os TT no Brasil foi montada. Experimentos realizados mostram que o uso de entropia para medir a frequência de postagens e escrita dos *tweets* é eficaz, com taxa de acerto de cerca de 92% dos *bots* na base de dados utilizada.

O restante do artigo é organizado como segue: na Seção 2 são apresentados trabalhos relacionados. A Seção 3.1 descreve os atributos passíveis de serem empregados na detecção de contas automatizadas, bem como os seis novos atributos propostos baseados em entropia. A Seção 4 apresenta a base de dados coletada e o processo de refinamento aplicado para se obter contas de humanos e automatizadas para a realização de experimentos. Na Seção 5 são mostrados os resultados dos classificadores testados com o conjunto de atributos utilizados. Por fim, na Seção 6 são expostas as conclusões obtidas e as possibilidades de trabalhos futuros.

2. Trabalhos Relacionados

Os trabalhos que buscam identificar atividade automatizada preocupam-se em determinar se os usuários são *bots* ou humanos, a fim de dar maior credibilidade aos dados que são utilizados para a construção de novas aplicações e serviços baseados no Twitter.

Para determinar se uma conta no Twitter possui comportamento automatizado, Zhang e Paxson [Zhang and Paxson 2011] analisaram as atualizações de contas de

usuários, utilizando apenas a informação do horário de publicação (disponível publicamente) e a fonte de onde o *tweet* foi publicado. Foram avaliadas 106.573 contas distintas, coletadas durante 3 semanas em abril de 2010. Os autores aplicaram um teste chi quadrado de Pearson[2], e descobriram que contas automatizadas exibem padrões de intervalos de atualizações em intervalos de tempo uniformes, que dificilmente podem ser observados para usuários humanos. Também perceberam que uma parcela dessas contas publicam seus *tweets* diretamente da Web (via página) e que palavras relacionadas a spam estão associadas com contas que possuem um alto grau de automação.

Freitas et al. [Freitas et al. 2014] abordaram o problema de detectar *bots* no Twitter focando na identificação de comportamento que foge às estratégias de identificação de atividade automática. Para tanto, utilizaram como base o teste de atividade automática aplicado por Zhang e Paxson [Zhang and Paxson 2011] em um conjunto de contas suspensas do Twitter, identificadas por [Ghosh et al. 2012], e em um conjunto de contas não-suspensas, totalizando 110.233 usuários. Identificando atributos linguísticos nas postagens dos usuários e padrões de comportamento capazes de diferenciar usuários *bots* e humanos, os autores foram capazes de detectar cerca de 92% dos *bots* na base de dados.

John et al. [Dickerson et al. 2014] propõem uma nova abordagem utilizando análise de sentimento nos *tweets*. Para tanto, um conjunto de atributos identificados por meio da ferramenta SentiMetrix's, juntamente com características já utilizadas em outras abordagens (conteúdo do *tweet* e comportamento do usuário, por exemplo), foram utilizadas para identificar *bots* no Twitter. Os autores demonstraram que o número de fatores relacionados a sentimento são a chave para a identificação de *bots*.

Neste artigo, como a intenção restringe-se a detecção de atividade automatizada nos tópicos de tendência do Twitter no Brasil, são utilizados atributos das contas e do conteúdo dos *tweets*, bem como também seis novos atributos, baseados no conceito de entropia, para medir padrões de escrita dos *tweets* e seu poder discriminativo para a tarefa de identificar atividade automatizada.

3. Atributos Extraíveis do Twitter

A forma como os usuários escrevem seus *tweets* e interagem com outros usuários pode ajudar a diferenciar comportamentos automatizados e não automatizados. Esta seção descreve o conjunto de atributos passíveis de serem empregados na detecção de contas automatizadas. Primeiro, são descritos os atributos de conteúdo e de comportamento. Em seguida, os seis novos atributos baseados em entropia são detalhados e uma discussão sobre sua capacidade de detecção de contas automatizadas em conjunto aos outros atributos é feita.

3.1. Atributos de Conteúdo e de Comportamento

Atributos de conteúdo são aqueles baseados em propriedades do texto dos tweets postados pelos usuários. Eles identificam aspectos relacionados com a forma como os usuários escrevem seus *tweets*, tipicamente através da análise da ocorrência (quantidade) de diversos atributos no texto. É importante ressaltar que outros atributos podem ser avaliados em termos da sua existência ou não. O conjunto de

[2]É um teste não paramétrico de hipóteses que se destina a encontrar um valor da dispersão para duas variáveis nominais, avaliando a associação existente entre variáveis qualitativas.

atributos de conteúdo dos *tweets* mais comuns são: *source_tweet*, *retweet_count*, *diversidade_lexica*, *media_tweets_dia*, *media_tweets_hora*, *num_palavras*, *media_palavras_tweet*, *media_urls_tweet*, *media_urls_topico*, *media_tweets_topico*, *media_palavras_topico*, *media_hashtags_topico*, *media_hashtags_tweet*, *media_mencao_tweet*, *media_mencao_topico*, *favorite_count_tweet*, *favorited_tweet* e *retweeted*.

Já os atributos de comportamento são aqueles baseados em propriedades do comportamento do usuário, em termos de frequência de postagens, interações sociais e influência, que procuram identificar características que tenham relação com a forma como os usuários interagem e sua popularidade no Twitter. Os atributos mais comuns de comportamento do usuário são: *id_user*, *verified*, *favourites_count_user*, *listed_count*, *protected*, *default_profile*, *num_followers*, *num_followed*, *rate_followers_followed*, *num_tweets*, *count_age* e *dif_tweets*.

Tanto os atributos de conteúdo quanto os de comportamento apresentados nesta seção podem ser encontrados nos trabalhos [Benevenuto et al. 2010, Lee et al. 2010, Stringhini et al. 2010, Wang 2010, Wang 2012] e todos já demonstraram serem eficientes para medir e diferenciar a intenção dos usuários, bem como caracterizar mensagem suspeitas ou envolvidas em atividades automatizadas (com objetivos maliciosos ou não).

3.2. Atributos baseados em Entropia

Neste artigo, a entropia de um conjunto de *tweets* postados por um usuário é calculada para medir o quanto de informação os *tweets* representam, levando em consideração todos os *tweets* que ele postou na base utilizada. Para tanto, seis novos atributos, extraídos das mensagens, foram propostos e são empregados para caracterizar comportamento automatizado (Tabela 1).

Tabela 1. Atributos Propostos Baseados em Entropia

Atributos	Descrição
entropia_total	Entropia total que representa o vocabulário de um usuário
media_entropia_tweets	Entropia média por *tweet* publicado
media_entropia_topico	Entropia média por tópico
entropia_usuario_topicos_diferentes	Entropia dos *tweets* de um usuário em tópicos diferentes, levando em consideração apenas as *hashtags* utilizadas
entropia_usuarios_mesmo_topico	Entropia dos *tweets* de usuários diferentes nos 10 tópicos com maior número de *tweets*
entropia_usuarios_topicos_diferentes	Entropia apenas para um *tweet* por tópico

O primeiro atributo, $entropia\_total$, representa a medida geral de entropia para o vocabulário de cada usuário, onde o vocabulário é entendido como todos os símbolos utilizados por ele nos *tweets* coletados e os símbolos são todas as palavras e caracteres obtidos de um dado conjunto de *tweets* postados pelo usuário. É importante ressaltar que, no que tange à implementação dessa característica, para cada *tweet* ou conjunto de *tweets*, uma lista com todos símbolos é gerada. Já os atributos $media\_entropia\_tweets$ e $media\_entropia\_topico$ representam, respectivamente, o valor do atributo $entropia\_total$ dividido pelo total de *tweets* que o usuário possui no conjunto de dados e o valor do atributo $entropia\_total$ dividido pelo número de tópicos que o usuário participou no conjunto de dados.

O atributo $entropia\_usuario\_topicos\_diferentes$ calcula, para cada usuário, a entropia do conjunto de *tweets* por tópico, levando em consideração apenas as *hashtags* utilizadas. Assim, é formado o conjunto de palavras chaves para cada usuário. Devido ao tamanho limitado dos *tweets*, muitos *bots* constroem mensagens utilizando apenas *hashtags* em seus *tweets* [Freitas et al. 2014]. Essa prática mostra uma estratégia para alcançar diversos tópicos de uma só vez.

O atributo $entropia\_usuarios\_mesmo\_topico$ calcula a entropia dos *tweets* de cada usuário para os 10 tópicos com maior número de *tweets*. Essa medida representa o quanto cada usuário pode variar o vocabulário participando do mesmo tópico. A intenção é medir o conteúdo dos *tweets* quando os usuários comentam sobre os mesmos assuntos nos tópicos com o maior número de *tweets*, já que são os mais representativos em número de mensagens.

O atributo $entropia\_usuarios\_topicos\_diferentes$ calcula a entropia apenas para um *tweet* por tópico. Essa medida mostra se os usuários escrevem *tweets* únicos ou com textos muito parecidos, independentemente do tópico. Ao escolher aleatoriamente um *tweet* por tópico, a intenção é medir se o usuário na base de dados possui um vocabulário realmente diversificado. A ideia é que para comentar sobre um mesmo tópico, normalmente um usuário poste mensagens não tão diversificadas, já que fazem referência a um mesmo assunto, enquanto que um *tweet* para cada tópico deve ser mais diversificado em termos de vocabulário, já que faz referência a assuntos diversos.

Todas as seis características propostas visam quantificar o quanto cada usuário diversificou as palavras e caracteres utilizadas em seus *tweets*. Assim, se o alfabeto de um usuário é bastante diversificado, então a entropia será elevada. Uma vez que usuários humanos postam mensagens com conteúdo variado, espera-se que possuam uma entropia maior em relação aos *bots* sociais que postam mensagens automáticas com conteúdo menos diversificado. Desta forma, em conjunto com os atributos de conteúdo e de comportamento, os novos atributos propostos devem ser capazes de mostrar características relevantes para diferenciar usuários humanos de usuários automatizados.

4. Base de Dados

A **base de dados inicial** empregada no desenvolvimento deste artigo foi formada por um total de 2.853.822 usuários e 11.294.861 *tweets* únicos, coletados no período entre dezembro de 2013 e junho de 2014. Durante esse período de coleta, foram encontrados 2.712 tópicos de tendência distintos (únicos) com cerca de 2.000.000 de URLs.

Dado o grande tamanho da base de dados, foi necessário utilizar algum mecanismo para reduzir a quantidade de usuários. Ao analisar a base de dados coletados, percebeu-se que embora muitos usuários ativos possuam grande número de mensagens publicadas no Twitter esses mesmos usuários publicam poucos *tweets* nos tópicos de tendência. Assim, após análises na literatura, optou-se por empregar o mecanismo implementado em [Zhang and Paxson 2011] para reduzir a quantidade de usuários. A ideia é que contas com menos de 30 *tweets* publicados durante o período de coleta foram consideradas insuficientes e foram descartadas por falharem no teste de atividade automática.

Por exemplo, o usuário X possui um total de 500 *tweets* (obtidos através do atributo $statuses\_count$) publicados no Twitter até o último dia de coleta, mas desse total

apenas 28 *tweets* foram publicados nos tópicos de tendência. Este usuário é considerado inválido por conter dados insuficientes para investigação de comportamento automatizado nos tópicos de tendência.

Após verificar o número de *tweets* de cada usuário no conjunto de dados, uma nova base de dados foi montada, contendo 50.012 usuários com mais de 30 *tweets* nos tópicos de tendência do Brasil. Em seguida, aplicando o teste de atividade automática aos horários e frequência de postagens por minuto, hora e dia, proposto por [Zhang and Paxson 2011], foram rotuladas 37.919 contas (aprovadas no teste de atividade automática) como usuários humanos e 12.093 contas foram reprovadas (pertencentes a contas de *bots*) rotuladas como bots. Para confirmar e assegurar a validade do teste, os perfis das contas classificadas como automatizadas e seus *tweets* foram consultados utilizando scripts que acessam os perfis dos usuários.

No final, a **base de dados final**, para treino e teste, é formada por 50.012 usuários (37.919 humanos e 12.093 automatizados), com 4.352.107 *tweets* e 829.082 URLs, sendo 322.918 URLs únicas. Assim, foi montada uma base de treinamento com 25.026 amostras rotuladas como Humano e 7.981 amostras rotuladas como Automatizado e uma outra base para teste contendo 12.893 amostras rotuladas como Humano e 4.112 amostras rotuladas como Automatizado. Essa distribuição de valores se refere a 64% das contas para treinamento e 36% para teste, como descrito na literatura de aprendizagem de máquina [Alpaydin 2010].

5. Avaliação e Resultados

Nesta Seção, o desempenho dos atributos discutidos na Seção 3.1, especialmente os seis novos propostos, são avaliados através de algoritmos de aprendizado de máquina supervisionada a fim de validar sua efetividade na tarefa de detectar *bots* sociais no Twitter. Na Seção 5.1, o processo de treinamento dos classificadores é descrito, incluindo as métricas de avaliação e os ajustes nos classificadores avaliados. Já na Seção 5.2, o teste de validação das novas características é apresentado e seus resultados discutidos.

5.1. Treinamento

O processo de treinamento é necessário para ajustes de parâmetros individuais ou em conjunto para cada classificador, a fim de obter o melhor desempenho de cada classificador sobre o conjunto de dados utilizado nesta fase. O resultado esperado é um modelo capaz de identificar comportamento automatizado utilizando os atributos propostos e, posteriormente, testar em uma nova base de dados para avaliar o desempenho dos classificadores.

No processo de treinamento de classificadores foram utilizados oito (08) classificadores. Contudo, serão apresentados somente os resultados dos quatro (04) melhores: Decorate, SVM, RandomForest e J48. Todos os classificadores foram testados usando a ferramenta Weka e utilizando o processo de validação cruzada de 10 partições para o treinamento.

Antes de demonstrar o melhor desempenho de cada classificador no processo de treinamento, é preciso informar as métricas de avaliação utilizadas. São elas: Precisão, Revocação, Medida F e Área sob a curva ROC (AUC). A Precisão é a porcentagem de amostras positivas classificadas corretamente sobre o total de amostras classificadas como positivas. A Revocação é a porcentagem de amostras positivas classificadas corretamente

sobre o total de amostras positivas. A Medida F é um indicador que combina a Precisão e a Revocação. Seu resultado está no intervalo [0,1], sendo o melhor resultado 1 e o pior resultado 0. AUC é um indicador usualmente adotado como medida de qualidade do classificador em aprendizagem de máquina [Sokolova et al. 2006].

5.1.1. Ajustes de Parâmetros e Escolha do Melhor Classificador

Para obter o melhor resultado sobre o conjunto de dados para cada classificador, foram realizados treinamentos (20.092 instâncias) onde os valores dos principais parâmetros de cada classificador foram ajustados até a obtenção do valor mais adequado. A Tabela 2 demonstra a comparação entre os resultados dos quatro classificadores, a fim de determinar o que melhor se ajusta ao conjunto de treinamento. Explicações sobre os parâmetros de ajuste, bem como sobre classificadores podem ser encontrados em [Alpaydin 2010].

Tabela 2. Comparação entre os Classificadores

Classificador	SVM	J48	Decorate	RandomForest
Parâmetros Ajustados	C=10, Kernel Polinomial Grau 1.0	FC=0,50	R=3.0 I=90	I=220 K=40
Precisão	76,90%	89,80%	89,70%	97,50%
Revocação	59,60,%	90,30%	92,80%	94,70%
Medida F	0.672	0.684	0.705	0.884
Área ROC	0.517	0.782	0.906	0.937

Como observado na Tabela 2, o classificador RandomForest foi o que obteve o melhor desempenho em relação aos demais classificadores.

A matriz de confusão (Tabela 3) ilustra o desempenho do RandomForest. Para a classe Automatizado, o modelo gerado acertou em torno de 92,40% das amostras no teste realizado para o melhor conjunto de parâmetros escolhidos no processo de treinamento. De igual modo, para a classe Humano, cerca de 94,73% das amostras foram classificadas corretamente, elevando assim o poder discriminativo do classificador testado.

Tabela 3. Matriz de confusão do RandomForest

RandomForest		Previsto	
		Humano	Automatizado
Correto	Humano	94,73%	5,27%
	Automatizado	7,60%	92,40%

Além disso, o RandomForest apresentou taxas de falso positivo de 5,30% para classe Humano e 7,60% para classe Automatizado. Assim, após o treinamento realizado, o RandomForest demonstrou ser o mais adequado para o problema de detectar e distinguir usuários humanos e automatizados na base de dados utilizada neste artigo, juntamente com o conjunto de atributos sugeridos.

5.2. Testes

A fim de avaliar o poder discriminativo para detectar comportamento automatizado, o classificador RandomForest que obteve os melhores resultados, foi testado em uma nova base de dados rotulada com 12.893 usuários humanos e 4.112 usuários automatizados.

5.2.1. Relevância dos atributos

A fim de medir a relevância dos atributos utilizados foi calculado o ganho de informação, isto é, o quanto cada atributo é representativo para o problema de distinguir usuários humanos e usuários automatizados. A Tabela 4 apresenta o ranking do ganho de informação dos 36 atributos empregados neste trabalho. Vale ressaltar que esses atributos foram apresentados na Seção 3.1.

Tabela 4. Ranking Ganho de Informação

Num	Atributo	Num	Atributo
1	media_tweets_hora	2	media_tweets_dia
3	media_tweets_topico	4	media_entropia_tweets
5	media_hash_topico	6	entropia_usuario_topicos_diferentes
7	diversidade_lexica	8	media_palavras_tweet
9	media_palavras_topico	10	media_mencao_topico
11	num_palavras	12	media_hash_tweet
13	entropia_usuarios_mesmo_topico	14	media_entropia_topico
15	media_urls_tweet	16	entropia_usuarios_topicos_diferentes
17	media_urls_topico	18	media_menção_tweet
19	dif_tweets	20	statuses_count
21	idade_conta	22	entropia_total
23	listed_count	24	razao_foll_fri
25	followers_count	26	friends_count
27	favourites_count_user	28	retweet_count
29	source_tweet	30	num_fontes
31	favorite_count_tweet	32	verified
33	default_profile	34	protected
35	favorited_tweet	36	retweeted

Os três (3) atributos de maior relevância estão relacionados a média de *tweets* postados por hora ($media\_tweets\_hora$), por dia ($media\_tweets\_dia$) e por tópicos ($media\_tweets\_topico$). Os dois primeiros ajudam a provar que usuários humanos compartilham mensagens de maneira não uniforme tanto por hora quanto no decorrer de um único dia, uma vez que as pessoas comentam sobre um tópico específico. Por outro lado, as contas automatizadas comentam em diversos tópicos de forma aleatória e mantém uma quantidade uniforme de mensagens postadas por faixa horária. No caso de postagens por tópico ($media\_tweets\_topico$), os *bots* normalmente participam ativamente em inúmeros tópicos enquanto os usuários humanos comentam de maneira esporádica, as vezes concentrando-se em poucos tópicos.

Os atributos relacionados à forma como os usuários escrevem seus *tweets* também demonstram ter uma grande relevância. Atributos como a média de *hashtags* por tópico ($media\_hash\_topico$) e por *tweet* ($media\_hash\_tweet$) demonstram que os *bots* abusam do uso de palavras chaves para marcar seus *tweets* e assim têm maior chances de serem encontrados, por exemplo, em uma busca por assuntos. A ideia é fazer com que um mesmo *tweet* esteja ligado a vários tópicos de uma só vez, alcançando assim um maior número de usuários, ganhando maior visibilidade e, por consequência, aumentando a possibilidade de alcançar mais seguidores. Do mesmo modo, usuários *bots* compartilham muitas URLs em suas mensagens e também fazem menção a outros usuários da rede. O uso extensivo de URLs pode ser uma forma de tentar atrair os usuários mencionados e os seguidores desses usuários a clicar nos links, que em muitos casos direcionam para conteúdo externo como vídeos, imagem ou outros sites.

Os atributos baseados no cálculo da entropia das mensagens demonstram ser eficazes e importantes para distinguir os usuários. Observando que os atributos relativos à média de entropia por *tweet* e por tópico estão entre os 20 principais atributos, é possível assumir que existe uma alto grau de similaridade entre mensagens de usuários *bots* para tópicos diferentes. Isso fica muito evidente em um busca na base de dados, onde o padrão de escrita destas mensagens normalmente abusa do uso de múltiplas hashtags muitas vezes só alternando a ordem destas na mensagem fazendo com que a mesma mensagem esteja presente em diversos tópicos.

Consequentemente a medida da entropia destas mensagens é praticamente a mesma já que o texto é praticamente o mesmo. No Twitter não é permitido publicar a mesma mensagem mais de uma vez em um curto intervalo de tempo, por isso os *bots* utilizam diversas *hashtags* ou invertem a ordem em que elas aparecem na mensagem para dar a ilusão de que são mensagens diferentes. Também utilizam a lista de tópicos para inserir alguns na mensagem e alternar entre esses tópicos, criando diversas mensagens similares. Uma análise na base de dados permite observar que usuários humanos compartilham mensagens com conteúdo variado.

5.2.2. Análise da Relevância dos Atributos Baseados em Entropia

Como forma de provar a importância dos atributos baseados em entropia, os quatro atributos mais bem ranqueados na subseção 5.2.1 foram avaliados usando 2.000 amostras, sendo 1.000 amostras de cada classe (Humano e Automatizado). Vale explicar que ao analisar diversas amostras da base final, contendo 2.000, 4.000 e 8.000 contas, percebeu-se a existência de uma similaridade nos resultados obtidos. Assim, optou-se por um conjunto de 1.000 amostras para cada classe, uma vez que processar a base final por completo (todas as amostras), de uma única vez, demanda um maior poder computacional e tempo. Os valores da entropia calculada para cada atributo variam de 0 a 6, e para fins de demonstração considera-se os valores no intervalo de [0,1) como entropia muito baixa, de [1,2) como entropia baixa, de [2,3) como entropia média baixa, de [3,4) como entropia média alta, de [4,5) como entropia alta e de [5,6] como entropia muito alta.

Entropia por *Tweet* e por Tópico

A Figura 1 demonstra que a entropia dos usuários humanos por *tweet* é medida como média e alta, enquanto usuários automatizados ficam na faixa entre muito baixa até média. Embora ambos os valores para as duas classes atinjam cerca de 80% dos *tweets* avaliados, os dados reafirmam que humanos possuem um vocabulário mais diversificado do que os automatizados. Além disso, os *bots* sociais tendem a ser mais repetitivos e menos espontâneos na escrita dos *tweets*.

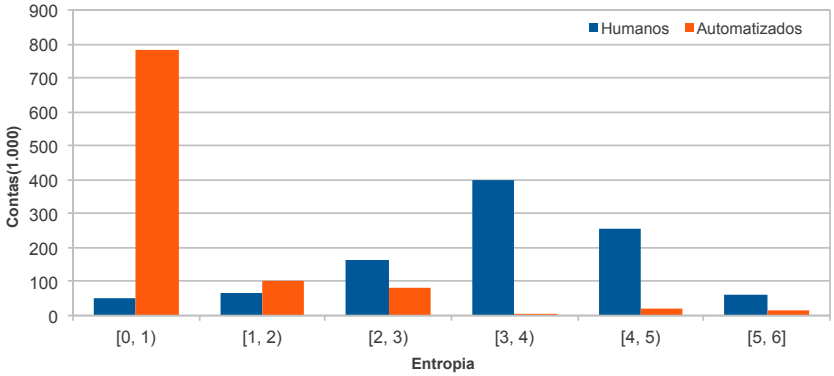


Figura 1. Entropia dos *Tweets*

Entropia dos Tópicos

De igual modo, a Figura 2 mostra que a entropia para o total de tópicos comentados é semelhante à entropia por *tweet*, uma vez que os *tweets* fazem referência aos tópicos.

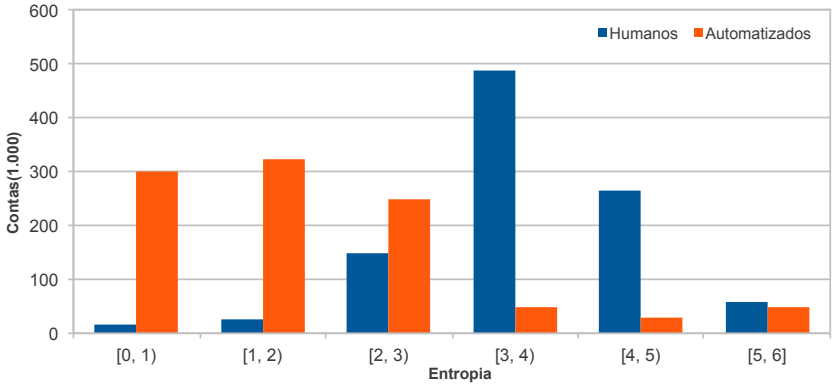


Figura 2. Entropia dos Tópicos

Entropia nos mesmos Tópicos e em Tópicos Diferentes

Na Figura 3 é possível notar que os usuários humanos, cerca de 87% que comentam em um mesmo tópico, possuem entropia considerada de baixa até média/alta, enquanto usuários automatizados, cerca de 82%, possuem entropia na faixa entre muito baixa e baixa. Usuários humanos que comentam várias vezes em um mesmo tópico não chegam a diversificar tanto o vocabulário, já que o assunto de todos seus *tweets* será específico para o tópico em questão. Por outro lado, os usuários automatizados postam

tweets massivamente para elevar a popularidade de um ou mais tópicos e acabam por repetir *hashtags* e URLs de maneira frequente nos *tweets*.

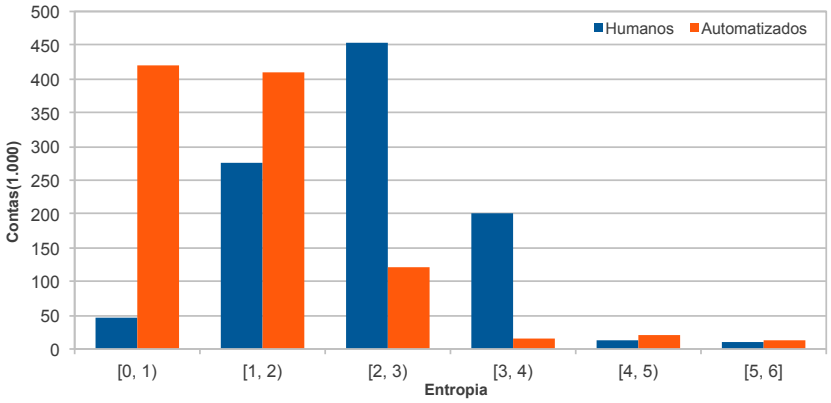


Figura 3. Entropia no mesmo Tópico

Entropia em Tópicos Diferentes

Na Figura 4, para *tweets* publicados em tópicos diferentes, nota-se que os usuários humanos, cerca de 94%, possuem entropia considerada alta ou muito alta, enquanto os usuários automatizados, cerca de 95%, possuem entropia considerada muito baixa a média. Isso reflete a maior divergência em termos de vocabulário, uma vez que, mesmo para tópicos diferentes, usuários automatizados publicam *tweets* com pouca diversidade de palavras, enquanto humanos escrevem de maneira natural e espontânea.

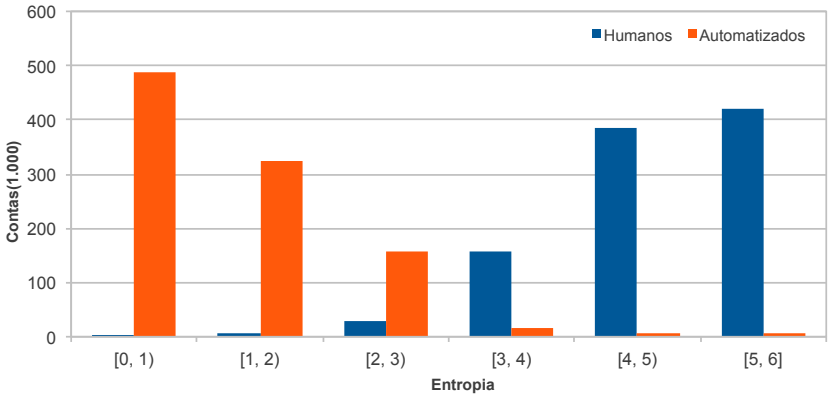


Figura 4. Entropia em Tópicos diferentes

5.3. Redução do Conjunto de Atributos

A fim de avaliar o poder discriminativo dos três (3) conjuntos de atributos separadamente, foram testados os atributos do conteúdo (C), do usuário (U) e de entropia (E) no classificador RandomForest. A Tabela 5 apresenta o desempenho obtido para cada um destes conjuntos ou ainda para mais de um conjunto.

Os resultados para o conjunto de atributos de conteúdo são os melhores, conseguindo identificar a maior parte dos *bots* sociais em relação aos outros conjuntos de

Tabela 5. Resultado do RandomForest

Atributos	Humanos	Automatizados
C	89,80%	86,95%
U	79,90%	82,90%
E	86,20%	83,00%
C+U	91,80%	91,70%
C+E	92,80%	92,10%
U+E	91,80%	90,50%
C+U+E	94,73%	92,40%

atributos. Em seguida está o conjunto de atributos baseados na entropia do texto das mensagens e por último os atributos do usuário.

Os atributos de conteúdo juntamente com os atributos de entropia representam o melhor resultado, ou seja, revelam que a forma como os usuários escrevem seus *tweets* é a principal característica para distinguir humanos de *bots*. O motivo é simples, esses dois conjuntos de atributos analisam justamente a forma como os *tweets* são escritos, quanto de similaridade possuem entre si e também o quanto a entropia do texto dos *tweets* é representativa para cada usuário.

Os atributos de usuários mostram-se, nesta escala, com menor poder discriminativo tanto isoladamente quanto em conjunto com outra categoria de atributos. Isso se deve ao fato principalmente de que muitos atributos do usuário não fazem distinção entre as classes humanos e *bots*, justamente porque na base final existem muitos *bots* ativos que participam normalmente dos tópicos de tendências postando notícias e atualizações de blogs e sites externos.

Comparando os resultados dos grupos de características avaliados neste artigo (conteúdo, usuários e entropia) com os alcançados no trabalho de [Freitas et al. 2014] (grupos usuário, conteúdo e linguístico), ambos acertando um pouco mais de 92% na identificação de *bots* sociais, percebe-se que a maneira como essas contas automatizadas escrevem seus *tweets* é fator determinante para sua detecção e caracterização. No caso da avaliação deste artigo, feita sobre os tópicos de tendência do Brasil, o uso da entropia para medir a frequência de postagens e escrita dos *tweets* demonstra ser eficaz na diferenciação de *bots* e usuários comuns.

6. Conclusão

Esta artigo apresentou seis (6) novas características, baseadas no conceito de entropia, aplicadas a detecção de comportamento automatizado no Twitter. Essas novas características permitem: (i) medir e diferenciar o padrão de escrita; (i) mensurar o vocabulário dos usuários; (iii) inferir quão diversificado o vocabulário de um usuário é; (iv) e classificá-lo com comportamento automatizado ou não.

Os resultados, utilizando aprendizagem de máquina, mostram que a junção dos atributos de usuário e de conteúdo com os atributos propostos permitem detectar comportamento automatizado nos tópicos de tendência do Twitter no Brasil. Empregando o classificador RandomForest, alcançou-se 92,40% de usuários automatizados e 94,73% de usuários humanos classificados corretamente. É necessário ressaltar que os resulta-

dos apresentados são exclusivos para a base de dados utilizada neste trabalho, mas que o resultado é similar a outros trabalhos da literatura.

Além disso, este artigo apresentou um metodologia para implantação de um processo de detecção de comportamento automatizado nos tópicos de tendência do Twitter. A importância deste trabalho se deve ao fato de que embora o próprio Twitter proíba a postagem de *tweets* automatizados nos tópicos de tendência, tal prática tem se tornado cada vez mais comum, podendo simplesmente degradar a experiência dos usuários como também expô-los a ataques e atividades maliciosas.

Como trabalho futuro é necessário criar grupos de atributos cada vez mais robustos e que representem, com maior relevância, comportamento automatizado. Outro ponto a ser considerado é criar mecanismos de detecção online de comportamento automatizado e/ou malicioso.

Referências

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2nd edition.

Benevenuto, F., Magno, G., Rodrigues, T., and Almeida, V. (2010). Detecting spammers on twitter. In *Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*.

Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2012). Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *Dependable and Secure Computing, IEEE Transactions on*, 9(6):811–824.

Dickerson, J. P., Kagan, V., and Subrahmanian, V. S. (2014). Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2014, Beijing, China, August 17-20, 2014*, pages 620–627.

Freitas, C., Benevenuto, F., and Veloso, A. (2014). Socialbots: Implicações na segurança e na credibilidade de serviços baseados no twitter. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Gara, T. (2013). One Big Doubt Hanging Over Twitter's IPO: Fake Accounts. `http://goo.gl/c1cSR4`.

Ghosh, S., Viswanath, B., Kooti, F., Sharma, N. K., Korlam, G., Benevenuto, F., Ganguly, N., and Gummadi, K. P. (2012). Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 61–70, New York, NY, USA. ACM.

Kartaltepe, E., Morales, J., Xu, S., and Sandhu, R. (2010). Social network-based botnet command-and-control: Emerging threats and countermeasures. In *Applied Cryptography and Network Security*, volume 6123 of *Lecture Notes in Computer Science*, pages 511–528. Springer Berlin Heidelberg.

Lee, K., Caverlee, J., and Webb, S. (2010). Uncovering social spammers: Social honeypots + machine learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 435–442, New York, NY, USA. ACM.

Malware Bytes (2014). Twitter Phishing Spamrun: "Strange Rumors About You'. https://blog.malwarebytes.org/fraud-scam/2014/09/twitter-phishing-spamrun-strange-rumors-about-you/.

Martinez-Romo, J. and Araujo, L. (2013). Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Systems with Applications*, 40(8):2992 – 3000.

Nazario, J. (2009). Twitter-based Botnet Command Channel. https://asert.arbornetworks.com/twitter-based-botnet-command-channel/.

Neal Ungerleider (2015). Almost 10% of Twitter is Spam. http://www.fastcompany.com/3044485/almost-10-of-twitter-is-spam.

Orcutt, M. (2012). Twitter mischief plagues mexicos election. http://www.technologyreview.com/news/428286/twitter-mischief-plagues-mexicos-election/.

Sokolova, M., Japkowicz, N., and Szpakowicz, S. (2006). Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, AI'06, pages 1015–1021, Berlin, Heidelberg. Springer-Verlag.

Stringhini, G., Kruegel, C., and Vigna, G. (2010). Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 1–9, New York, NY, USA. ACM.

Symantec (2013). Phishing: The Easy Way to Compromise Twitter Accounts. http://goo.gl/KJfu39.

Twitter (2014). Denunciar spam no Twitter. https://support.twitter.com/articles/263349-como-denunciar-por-spam-no-twitter.

Wang, A. (2012). Machine learning for the detection of spam in twitter networks. In *e-Business and Telecommunications*, volume 222 of *Communications in Computer and Information Science*, pages 319–333. Springer Berlin Heidelberg.

Wang, A. H. (2010). Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–10.

Zhang, C. M. and Paxson, V. (2011). Detecting and analyzing automated activity on twitter. In *Proceedings of the 12th International Conference on Passive and Active Measurement*, PAM'11, pages 102–111, Berlin, Heidelberg. Springer-Verlag.

Recomendações de Características Ergonômicas para Interfaces de Sistemas de Monitoramento de Redes Baseadas em Critérios de Usabilidade

Wilma E. Rosado[1], Leobino N. Sampaio[2], José A. S. Monteiro[3]

[1]Instituto Federal da Bahia (IFBA)
Camaçari – BA – Brasil

[2]Programa de Pós-graduação em Ciência da Computação (PGCOMP)
Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

[3]Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brasil

wilmaedysley@gmail.com, leobino@ufba.br, suruagy@cin.ufpe.br

Abstract. *Since they have been designed by experts, the interfaces of some of the current network monitoring systems have neglected important usability aspects for non-expert users. To assist designers in developing network interfaces more appropriated to non-expert users, this paper presents a set of recommendations of ergonomic features built according to usability criteria. Furthermore, an interface evaluation methodology is proposed in order to provide a quantitative measure of the interfaces adequacy. The effectiveness of the recommendations and the evaluation methodology were verified by analyzing the PerfsonarUI tool and a set of monitoring tools available in CAIDA.*

Resumo. *Por serem projetadas por especialistas, as interfaces de alguns dos atuais sistemas utilizados no monitoramento de redes possuem aspectos importantes de usabilidade negligenciados para usuários leigos. No intuito de auxiliar os projetistas no desenvolvimento de interfaces de sistemas de monitoramento de redes mais adequadas a usuários não especialistas, este artigo apresenta um conjunto de recomendações de características ergonômicas criadas a partir de critérios de usabilidade. É ainda proposta uma metodologia de avaliação das interfaces com o objetivo de apresentar uma medida quantitativa de adequação das interfaces. A aplicabilidade das recomendações e da metodologia de avaliação propostas foram verificadas por meio da análise da ferramenta PerfsonarUI e um conjunto de ferramentas de monitoramento disponíveis no CAIDA.*

1. Introdução

Ao longo dos anos, as iniciativas voltadas para o monitoramento de redes estiveram essencialmente focadas na instrumentação das infraestruturas de comunicação; na coleta e no armazenamento de dados [Bajpai and Schönwälder 2015]. Apesar de alguns avanços recentes na área, tanto usuários avançados quanto equipes de suporte ainda

contam com aplicações de monitoramento que apresentam informações sobre o desempenho de redes a partir de uma perspectiva puramente técnica. Esse é um entendimento crescente na comunidade de pesquisa. Em recentes Workshops do PerfSONAR [Swany and Calyam 2014], foi identificada a necessidade de que sejam desenvolvidos recursos e facilidades cobrindo casos de uso das ferramentas de monitoração por usuários não especialistas. Dada a diversificação de perfis de usuários interessados em informações de desempenho das redes, foi constatada a importância do desenvolvimento de interfaces de sistemas de monitoração que não somente apresentem as informações de desempenho, mas também facilitem o acesso e a compreensão das informações provenientes das medições. Por tais motivos, as técnicas de visualização da informação, avaliadas a partir de uma perspectiva centrada no humano, passa a ser um dos próximos desafios na área [Guimarães et al. 2016].

Na tentativa de facilitar a manipulação das informações de desempenho de redes, algumas infraestruturas de monitoramento têm apresentado recursos mais avançados e amigáveis de visualização [Guimarães et al. 2016, Legrand et al. 2009, Oberheide et al. 2006, Hanemann et al. 2006], contudo, são desenvolvimentos pontuais. Na prática, as características das interfaces de acesso às informações de monitoração são decorrentes de percepções informais de usuários especialistas em redes ou desenvolvedores que não seguem metodologias formais de desenvolvimento. A construção de interfaces a partir de critérios de usabilidade, frequentemente discutidos pela comunidade de IHC (Interação Humano–Computador), tem sido pouco explorada pela comunidade de gerência de redes e operação de serviços, ainda que os recentes avanços da tecnologia tenham permitido a evolução dos métodos e técnicas de avaliação de interfaces para o desenvolvimento de sistemas. O uso de técnicas de avaliação e desenvolvimento de interfaces de sistemas discutidas sobre IHC é muito amplo e certamente o extenso número de critérios, de caráter subjetivo, dificulta a adesão por parte da comunidade de redes. Como resultado, as interfaces dos sistemas apresentam características de usabilidade que pouco beneficiam usuários leigos[1].

Por tais motivos, este artigo apresenta um conjunto de recomendações de características ergonômicas para interfaces de sistemas utilizados no monitoramento de redes. As recomendações são baseadas em critérios de usabilidade e buscam oferecer subsídios para a realização de uma avaliação mais objetiva. Além das recomendações, este trabalho apresenta as seguintes contribuições: (i) a proposta de uma metodologia que busca apresentar uma métrica quantitativa na avaliação de interfaces de monitoramento. Esta métrica refere-se ao grau de adequação de uma interface a um perfil de usuários; (ii) a análise de um grupo de oito ferramentas utilizadas pela comunidade de gerência e operação de redes nas atividades de monitoramento. Esta análise permitiu identificar problemas de usabilidade de algumas ferramentas de monitoramento, assim como os níveis de adequação aos usuários não especialistas. A ferramenta PerfsonarUI [Jeliazkova et al. 2006], em particular, foi escolhida para a apresentação de um caso de uso completo.

Este artigo está organizado da seguinte forma. A Seção 2 discute brevemente os principais problemas das atuais interfaces de sistemas de monitoração para usuários não especialistas. A Seção 3 apresenta as recomendações de características ergonômicas para interfaces de sistemas utilizados no monitoramento de redes, assim como a proposta de

[1]No contexto deste trabalho, este perfil corresponde aos usuários não especialistas em redes.

uma metodologia que, através das recomendações, busca apresentar uma métrica quantitativa de análise. A Seção 4 descreve o uso das recomendações propostas por meio de uma metodologia de avaliação de interfaces que foi utilizada na análise de interfaces de ferramentas de monitoramento existentes. Por fim, a Seção 5 apresenta as conclusões do trabalho.

2. Problemas das atuais interfaces de sistemas de monitoração

As recomendações propostas neste artigo foram elaboradas a partir de um levantamento de viés indicativo que teve como objetivo coletar dados referentes às dificuldades de usabilidade encontradas nas atuais aplicações de monitoramento de redes. O levantamento consistiu numa pesquisa de campo envolvendo a Avaliação Prospectiva [Rubin 1994] e procurou identificar os problemas de interface mais comuns de aplicações utilizadas no monitoramento de redes. No questionário aplicado, levou-se em consideração os critérios ergonômicos definidos por [Bastien and Scapin 1993], comumente utilizados durante a avaliação em testes de usabilidade para verificar a eficiência, efetividade e facilidade de uso das interfaces.

O questionário foi aplicado a um grupo de treze usuários, majoritariamente pertencentes a ambientes acadêmicos e de pesquisa oriundos de instituições de ensino no estado da Bahia[2]. A análise das respostas apresentadas pelos usuários permitiram identificar a ausência de alguns critérios ergonômicos nas interfaces dos sistemas utilizados pelos mesmos. Entre os pontos mais relevantes, foi identificado que *Agrupamentos de Itens por Localização e Formato*, que se referem ao destaque no posicionamento dos itens na interface, são perceptíveis ao usuário especialista, contudo não interferem na usabilidade dos sistemas. A ferramenta CACTI, por exemplo, foi citada como uma interface que "*não possui uma percepção visual consistente*". Foi percebido o mesmo fato com o critério *Ações Mínimas* que faz referência à quantidade mínima de ações necessárias para realizar uma tarefa. Já os critérios *Legibilidade* e *Concisão* destacaram problemas complexos que afetam a execução e interpretação dos dados. Esses critérios buscam garantir as características léxicas que as informações apresentam sobre a tela e, a permissão de entradas ou a eliminação de informações redundantes na aplicação, respectivamente. A legibilidade da interface não foi diferenciada entre ferramentas gráficas e textuais, ambas adotam os critérios com deficiência. Com relação à entrada de informações, algumas ferramentas foram destacadas por possuírem interfaces "*com risco de entrada de informações redundantes caso o usuário seja iniciante*", como o caso da MRTG. Por fim, usuários enfrentam problemas graves com os erros gerados pelas aplicações de monitoramento. Assim, resultados indicam para a necessidade de se priorizar ainda mais o tratamento de informações inseridas no sistema de modo a tratar redundâncias, como se refere o critério *Proteção Contra Erros*.

Maiores detalhes sobre os resultados obtidos e a metodologia utilizada no levantamento realizado para identificar os problemas das atuais interfaces de monitoração podem ser encontrados em [Rosado 2011].

[2]Universidade Salvador (UNIFACS) e Instituto Federal da Bahia (IFBA).

3. Recomendações de características ergonômicas

Esta seção apresenta as recomendações de características ergonômicas para interfaces de sistemas de monitoramento propostas neste artigo. As mesmas foram elaboradas a partir dos problemas identificados na pesquisa de campo descrita na Seção 2 e tiveram como base alguns dos critérios ergonômicos discutidos nas pesquisas sobre IHC.

De forma resumida, a metodologia adotada para a elaboração das recomendações foi: i) identificar os problemas de usabilidade apontados por usuários de interfaces de monitoramento de redes; ii) Fazer um levantamento de critérios ergonômicos de usabilidade; iii) mapear critérios ergonômicos aos problemas de usabilidade identificados; iv) elaborar as recomendações a partir dos critérios e considerando as especificidades das atividades de monitoramento de redes e características de usuários não especialistas. Ao final, as recomendações foram avaliadas a partir da análise de interfaces de monitoramento existentes. Os dados resultantes dessa análise são discutidos na Seção 4.

3.1. Critérios ergonômicos de usabilidade de interface de sistemas de computador

Conforme citado anteriormente, a metodologia utilizada para a elaboração das recomendações propostas consistiu em tomar como base os critérios ergonômicos comumente utilizados em avaliações de testes de usabilidade em sistemas computacionais. O objetivo foi de verificar a eficiência, efetividade e facilidade de uso das interfaces. Os critérios identificados como mais relevantes para as interfaces de sistemas de monitoramento foram: a) **Condução**, que se refere à organização dos itens na tela, a legibilidade dos mesmos e se há retorno expressivo e eficiente para o usuário final; b) **Carga de Trabalho**, que se faz importante, principalmente, pela memorização do acesso às funções da aplicação; c) **Controle Explícito**, onde defende-se o domínio das funções e liberdade ao usuário para realizar ações a partir dos seus conhecimentos técnicos e, por fim; d) **Gestão de Erros**, que prioriza um tratamento eficiente para que os erros gerados sejam bem interpretados pelo usuário da aplicação.

3.2. Recomendações de características ergonômicas para interfaces de aplicações de monitoramento de redes

O conjunto de recomendações foi elaborado a partir dos critérios e subcritérios definidos em [Bastien and Scapin 1993]. Assim, sete recomendações foram identificadas, a saber: *R1 – Adaptação de Características Léxicas e Imagens*, *R2 – Tratamento de Entrada de Informações*, *R3 – Utilização de Menus*, *R4 – Controle de Execuções Automáticas*, *R5 – Suporte ao Usuário*, *R6 – Controle do Uso de Linguagens Técnicas*, e *R7 – Padronização de Ambientes*. A Figura 1 resume a relação de tais recomendações e os critérios ergonômicos de usabilidade. As próximas subseções descrevem cada uma das recomendações propostas.

3.2.1. R1 – Adaptação de Características Léxicas e Imagens

A recomendação **Adaptação de Características Léxicas e Imagens** tem como objetivo inserir, no contexto da interface, fatores de origem perceptiva no intuito de proporcionar melhor visibilidade dos resultados provenientes das medições da rede. Cor, imagens e gráficos são alguns aspectos que facilitam a compreensão das informações apresentadas,

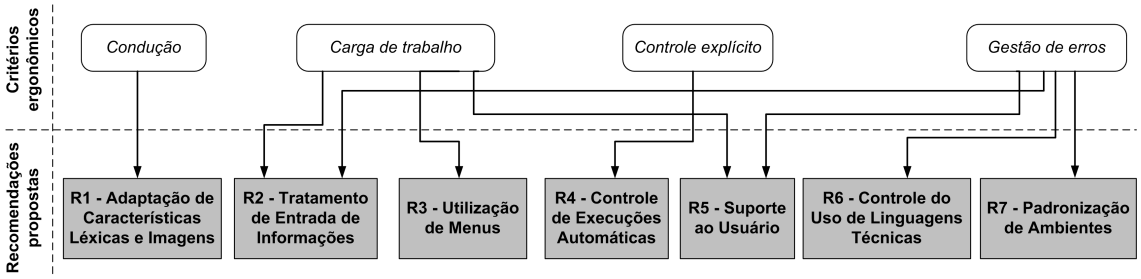


Figura 1. Critérios ergonômicos e as recomendações propostas.

sobretudo quando a ferramenta apresenta informações sobre as diferentes condições de tráfego. Como a maioria das ferramentas é projetada para usuários especialistas, na maioria das vezes as informações produzidas pelas ferramentas são apresentadas de forma textual, apenas com uso de números e abreviações. Além dos elementos de cor, imagens e gráfico é preciso destacar também a estrutura visual da interface. É recomendável o agrupamento de itens semelhantes, ou com funcionalidades similares que facilitam a compreensão do usuário. Tanto usuários especialistas quanto não especialistas priorizam um resultado consolidado ao invés de ter que visualizar e interpretar exaustivamente resultados parciais.

3.2.2. R2 – Tratamento de Entrada de Informações

O **Tratamento de Entrada de Informações** consiste numa recomendação voltada para a verificação das informações que são inseridas na aplicação de forma a impedir a redundância e minimizar a ocorrência de erros durante a execução das ações. Através dessa recomendação busca-se a concisão e proteção contra erros geralmente cometidos por usuários com pouco conhecimento sobre questões relacionadas ao desempenho das redes de computadores. Sistemas de monitoração frequentemente exigem um conjunto de parâmetros técnicos que normalmente o público geral não lida com muita frequência. Uma das estratégias comuns para evitar tais problemas consiste no tratamento e verificação da consistência das informações inseridas durante o uso da aplicação. Pode-se, por exemplo, evitar um erro de digitação de dois endereços IPs referentes à origem e destino de um caminho fim a fim para o qual o usuário deseja verificar o desempenho.

3.2.3. R3 – Utilização de Menus

A recomendação **Utilização de Menus** foi proposta com o objetivo de agrupar as funções disponibilizadas pela aplicação. O agrupamento organiza e reduz o tempo de realização de uma determinada ação do usuário final. Essa recomendação é resultante dos critérios *Ações Mínimas* e *Memorização*, definidos por Nielsen [Nielsen 1993], o qual afirma que as interfaces devem apresentar facilidade de memorização permitindo que usuários ocasionais consigam utilizá-las mesmo depois de um longo intervalo de tempo. Ferramentas de monitoramento fazem uso de ambientes textuais, o que dificulta ainda mais a utilização de recursos de menu, se tornando ainda menos adequadas ao grupo de usuários leigos.

3.2.4. R4 – Controle de Execuções Automáticas

A fim de possibilitar a intervenção do usuário nas funções executadas pelo sistema, por meio do cancelamento de operações, suspensão de tarefas, etc., foi proposta a recomendação **Controle de Execuções Automáticas**. A mesma refere-se, especificamente, ao critério de usabilidade *Controle Explícito* [Bastien and Scapin 1993], o qual ressalta a importância da liberdade de manipulação do usuário durante a realização das suas tarefas. A importância desse recurso para as ferramentas de monitoramento a usuários leigos deve-se ao fato dos mesmos frequentemente mudarem de opinião sobre uma determinada ação com a ferramenta. Esse recurso é bastante relevante, por exemplo, numa ferramenta que realiza testes de largura de banda sob demanda, uma vez que as técnicas utilizadas consomem uma razoável quantidade de recursos da rede (Ex. *Iperf*). Através de tais ferramentas, o usuário pode executar um teste inapropriado, necessitando a suspensão de execução devido a uma possível sobrecarga da rede.

3.2.5. R5 – Suporte ao Usuário

Dado que o objetivo desse trabalho busca beneficiar usuários não especialistas em temas de desempenho de redes, a recomendação **Suporte ao Usuário** torna-se uma das mais relevantes do conjunto proposto. A mesma verifica se as informações disponibilizadas na interface (referente às funções da aplicação) são suficientes para que o usuário possa interagir satisfatoriamente e sem auxílios. Comumente, aplicações fazem o uso do recurso *ajuda* como solucionador para a interpretação de funções e resolução de possíveis eventualidades. A questão é que este tipo de suporte não é suficiente, pois os usuários, em sua grande maioria, não exploram todas as informações disponíveis pelos recursos de ajuda. Nielsen [Nielsen 1993] relata este aspecto quando definiu os *slogans* de usabilidade para o critério "Help não ajuda". As informações são dispostas de forma complexa, comprometendo a interpretação. Assim, quanto menos o recurso *ajuda* for utilizado, melhor será o desempenho do usuário ao realizar suas tarefas. Em aplicações de monitoramento, essa questão é ainda mais relevante, dado que a dinâmica imposta pelas atividades de avaliação de desempenho das redes exige sistemas providos de recursos mais simples e efetivos de suporte ao usuário.

3.2.6. R6 – Controle do Uso de Linguagens Técnicas

A recomendação **Controle do Uso de Linguagens Técnicas** tem como objetivo facilitar a interpretação das informações disponibilizada pela aplicação de monitoramento de redes ao sugerir a apresentação das informações de desempenho em alto nível, sem tantos detalhes técnicos. O sistema deve, portanto, tratar das mensagens geradas na monitoração, de forma a garantir o entendimento do usuário por meio da utilização de linguagens menos técnicas. As mensagens de erro devem ser geradas de forma a garantir que o usuário compreenda qual a eventualidade ocorrida e qual procedimento deve ser realizado para que a aplicação retorne ao seu estado normal de execução. O uso de linguagem técnica deve ser dosado visto que o usuário leigo pode enfrentar dificuldades em decifrar códigos, mensagens de compilação ou execução que comumente são apresentadas pelas atuais aplicações de monitoramento de redes.

3.2.7. R7 – Padronização de Ambientes

A recomendação **Padronização de Ambientes** viabiliza a manipulação da interface, proporcionando ao usuário estruturas semelhantes a outras aplicações e facilitando também a memorização das ações. A frequência de exposições semelhantes entre as aplicações de monitoramento contribui para a eficiência da interação. Nielsen [Nielsen 1993] mais uma vez relata entre os seus atributos de usabilidade, a *Facilidade de Relembrar*. Este atributo avalia as funcionalidades da aplicação para que sejam fáceis de serem relembradas, mesmo que o usuário passe um longo período sem utilizá-la. Para este critério, a recomendação se enquadra no conceito de seguir os padrões de desenvolvimento de interfaces, propiciando um ambiente capaz de relembrar finalidades de funções também disponibilizadas por outras aplicações.

3.3. Contribuições de usabilidade para usuários não especialistas

A Tabela 1 resume as recomendações propostas e os benefícios esperados ao usuário não especialista em redes de computadores. É possível observar que as contribuições de usabilidade são complementares. Quando tais contribuições forem obtidas na sua totalidade, espera-se que o usuário final tenha uma interface com mais características ergonômicas e mais adequadas à realização de atividades de monitoramento de redes, mesmo quando não se tratar de um especialista da área.

Tabela 1. Recomendações propostas e principais contribuições de usabilidade aos usuários não especialistas.

Recomendação	Contribuições de usabilidade para o usuário não especialista
R1 – Adaptação de Características Léxicas e Imagens	Melhor visibilidade dos resultados provenientes das medições da rede por meio de elementos de Cor, imagens e gráficos; Agrupamentos de itens semelhantes que facilitam a compreensão do usuário.
R2 – Tratamento de Entrada de Informações	Evita a redundância e minimiza a ocorrência de erros durante o uso das aplicações de monitoramento.
R3 – Utilização de Menus	Organiza e reduz o tempo de realização de uma determinada ação do usuário final.
R4 – Controle de Execuções Automáticas	Liberdade de manipulação da ferramenta durante a sua execução.
R5 – Suporte ao Usuário	Informações de suporte disponibilizadas na interface sem necessitar requisição do usuário.
R6 – Controle do Uso de Linguagens Técnicas	Apresentação das informações de desempenho em alto nível e com reduzida quantidade de detalhes técnicos.
R7 – Padronização de Ambientes	Interfaces padronizadas que facilitam a memorização das ações.

4. Metodologia de avaliação de interfaces

O conjunto das recomendações descritas na seção anterior pode ser aplicado de diversas formas na avaliação e construção de interfaces de aplicações utilizadas no monitoramento de redes. Este trabalho também propõe uma metodologia de avaliação de interfaces que, a partir do nível de conformidade das interfaces com as recomendações propostas, apresenta uma medida de adequação da interface ao usuário final. Trata-se de uma medida quantitativa que permite uma avaliação mais objetiva das interfaces dos sistemas.

A metodologia está resumida na Figura 2. A Etapa 1 destina-se a fazer um levantamento do perfil dos usuários da ferramenta. O perfil auxilia na identificação das recomendações mais relevantes que, posteriormente, são utilizadas para a definição dos

pesos que definem o grau de importância. Na Etapa 2 são executadas duas atividades. Na primeira, são atribuídos pesos a cada recomendação a partir do grau de importância identificado. Os pesos podem ser atribuídos por meio de técnicas utilizadas na atribuição de valores quantitativos a critérios subjetivos, tais como a AHP (do inglês, *Analytic Hierarchy Process*) [Saaty 2008]. Já a segunda atividade consiste na avaliação da interface, a fim de identificar o nível de conformidade de cada recomendação. Por fim, a Etapa 3 consiste em utilizar as informações de pesos e conformidade das interfaces para calcular o grau de adequação da mesma ao usuário.

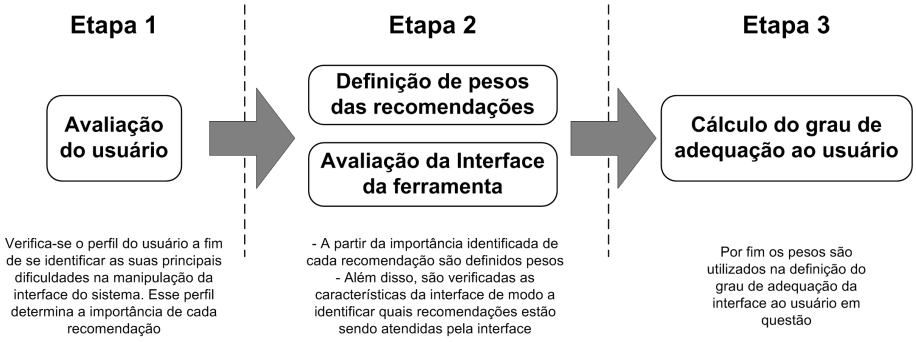


Figura 2. Metodologia de definição do grau de adequação das interfaces.

A aplicabilidade das recomendações propostas neste artigo foi verificada por meio da análise das interfaces de nove aplicações utilizadas no monitoramento de redes. As análises das interfaces foram realizadas seguindo a metodologia descrita nesta seção. Assim, a Subseção 4.1 descreve a implementação das etapas descritas na Figura 2. A Subseção 4.2 descreve um caso de uso completo feito através da ferramenta PerfSONAR UI e que descreve como todas as etapas da metodologia foram implementadas. Em seguida, a Subseção 4.3 apresenta os resultados das análise feita com outras oito ferramentas de monitoramento, disponibilizadas no site do CAIDA (*Center for Applied Internet Data Analysis*)[3].

4.1. Implementação das etapas

A implementação das etapas foi feita no intuito de verificar o grau de adequação das interfaces de um grupo de nove ferramentas, conforme a análise das interfaces descritas nas Seções 4.2 e 4.3. O perfil dos usuários (Etapa 1) foi identificado a partir da pesquisa de campo, descrita na Seção 2. Para a definição dos pesos foi adotada a técnica de comparação pareada, enquanto que a avaliação das interfaces dependeu de cada ferramenta analisada (Etapa 2). Por fim, o cálculo do grau de adequação (Etapa 3) foi obtido a partir de uma média ponderada das notas de conformidade encontradas ponderada pelos pesos das recomendações.

4.1.1. Etapa 1: Perfil do usuário

A caracterização do perfil do usuário foi realizada a partir do levantamento descrito na Seção 2. Em sua grande maioria, os dados apontam que tais usuários manipulam

[3]http://www.caida.org

exaustivamente, e muitas vezes erroneamente, a interface à procura de clareza para as interpretações de dados de desempenho da rede. Isso se deve ao fato dessas aplicações assumirem que o usuário final possui conhecimento técnico satisfatório para interpretar quaisquer dados de medições, geralmente encontrado nos especialistas (ex. gerentes e operadores). Além disso, o estudo de campo também identificou que usuários leigos buscam recursos gráficos nas interfaces que os auxiliem na compreensão dos dados. O nível técnico da linguagem utilizado para expressar informações da rede, o padrão do ambiente entre as ferramentas e a organização dos itens da interface da aplicação também foram identificados como aspectos essenciais para o perfil do usuário leigo.

4.1.2. Etapa 2: Definição de pesos das recomendações e avaliação da interface

Para a identificação dos pesos das recomendações utilizados no cálculo da adequação da interface foi adotada a técnica de comparação pareada utilizada no Processo Analítico Hierárquico AHP (do inglês, *Analytic Hierarchy Process*) [Saaty 2008]. A comparação pareada permitiu a atribuição de valores numéricos à cada recomendação de modo a refletir a importância das mesmas para o perfil do usuário leigo. O procedimento consistiu na realização de uma avaliação sistemática das recomendações por meio da comparação em par das mesmas. Os critérios adotados nesta comparação foram definidos a partir do perfil do usuário identificado na Etapa 1.

A Figura 3 (a) apresenta a matriz final utilizada na comparação pareada entre as recomendações. Cada elemento a_{ij} da matriz é definido a partir da predominância, em importância, de uma recomendação i sobre a outra j e vice-versa. Em casos de recomendações com importâncias iguais, ambas as recomendações i e j são atribuídas à posição a_{ij}. O peso de uma recomendação R é então definido a partir do percentual de vezes que a mesma aparece na matriz. A Figura 3 (b) apresenta o percentual obtido e utilizado na definição dos pesos. Ou seja, foram obtidos os pesos $19,2; 3,8; 11,5; 3,8; 19,2; 23,1$ e $19,2$ para as recomendações R1 a R7, respectivamente.

(a)

	R1	R2	R3	R4	R5	R6	R7
R1		*R1*	*R1,R3*	*R1*	*R5*	*R1, R6*	*R1*
R2			*R3*	*R4*	*R2, R5*	*R6*	*R7*
R3				*R3*	*R5*	*R6*	*R7*
R4					*R5*	*R6*	*R7*
R5						*R5, R6*	*R7*
R6							*R6, R7*
R7							

(b)

Rec	Qtd	Peso
R1	5	19,2%
R2	1	3,8%
R3	3	11,5%
R4	1	3,8%
R5	5	19,2%
R6	6	23,1%
R7	5	19,2%
Total	26	100%

Figura 3. Definição dos pesos das recomendações através da comparação pareada.

No que diz respeito à avaliação da interface da ferramenta corresponde a uma atividade da Etapa 2 que busca identificar os níveis de conformidade das recomendações (C_R) da interface do sistema de monitoramento. Na abordagem adotada nessa avaliação foram atribuídas notas de conformidade com valores entre 0 a 10 para cada uma das recomendações avaliadas na ferramenta de monitoramento.

4.1.3. Etapa 3: Cálculo da adequação da interface ao usuário

Na Etapa 3 utilizamos as notas de conformidade de cada recomendação atribuídas à interface da psUI em conjunto com os respectivos pesos, conforme apresentado na Equação 1. Através dessa equação nós buscamos obter um valor quantitativo, em termos percentuais, que traduz o nível de adequação da interface da ferramenta ao usuário final. Assim, denota-se por $A\,(u)$, o nível de adequação total da interface para um usuário u. O valor é resultante da média ponderada das notas de conformidade C_{R_i} da recomendação R_i verificada na interface do sistema de monitoramento avaliado, onde $i = \{1...7\}$, uma vez que existe um total de sete recomendações propostas. É preciso lembrar que o peso de cada recomendação (W_{R_i}) varia de acordo com o perfil do usuário e a sua relevância, conforme discutido nas seções anteriores.

$$A\,(u) = \frac{\sum_{i=1}^{7} C_{R_i} \times W_{R_i}}{\sum_{i=1}^{7} W_{R_i}}. \tag{1}$$

4.2. Avaliação da Ferramenta PerfSONAR UI

A PerfSONAR UI (psUI) [Jeliazkova et al. 2006] consiste em uma das ferramentas utilizada pelo serviço de monitoramento da rede europeia Géant2. Trata-se de ambiente Web desenvolvido para executar e visualizar testes sob demanda entre os pontos de medição do ambiente PerfSONAR [Hanemann et al. 2005], ou ainda visualizar resultados a partir dos arquivos de medição (MAs). A sua interface foi desenvolvida com o objetivo de ser flexível ao usuário final, permitindo que os possíveis problemas da rede sejam solucionados com rapidez e eficiência.

Para a avaliação descrita nesta seção, foi utilizada a versão online[4] da psUI disponibilizada pelo serviço de medições da rede Géant2. O acesso se deu através da autenticação federada por meio da Comunidade Acadêmica Federada (CAFe)[5] através de um usuário cadastrado na UFBA (Universidade Federal da Bahia). Por fim, é preciso destacar que a escolha da psUI deveu-se ao fato de ser uma ferramenta que está sendo utilizada atualmente pela comunidade acadêmica, vinculada não somente à Géant2, mas também à Internet2 nos EUA e à RNP (Rede Nacional de Ensino e Pesquisa) no Brasil. Esta subseção descreve apenas as especifidades das etapas adotadas na análise da ferramenta PersonarUI, uma vez que os procedimentos adotados já foram descritos na Subseção 4.1.

4.2.1. Análise da interface da ferramenta psUI

A Figura 4 apresenta o conjunto das principais interfaces da psUI utilizadas na avaliação. Em (a) está a tela inicial da ferramenta, dividida em duas partes: i) seleção e configuração,

[4]Acessivel através da url: http://ps-ui-test.qalab.geant.net/perfsonar-ui/

[5]https://portal.rnp.br/web/servicos/cafe

localizada ao lado direito da tela e, ii) a área de visualização dos resultados apresentados. Por meio da janela de seleção é possível acessar as funções **Access**, **Analyse** e **Settings**.

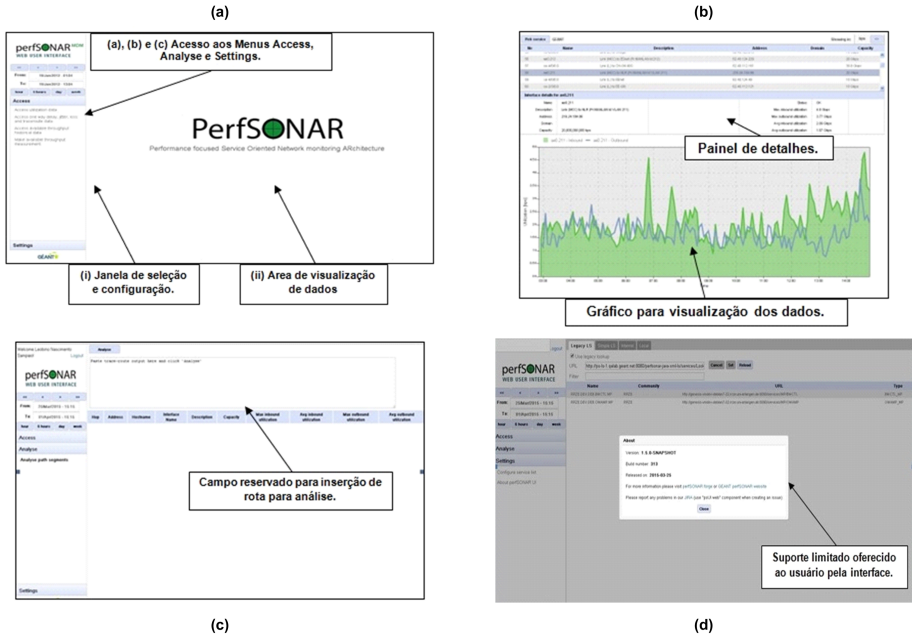


Figura 4. Principais interfaces da ferramenta PerfsonarUI.

A recomendação *Adaptação de Características Léxicas e Imagens* (R1) faz parte da estrutura da psUI. A Figura 4 (b) mostra que ela dispõe de uma visualização de dados por meio de gráfico, que facilita a interpretação dos dados de medição. O gráfico representa dados de utilização do *link* para uma determinada interface selecionada. Após a seleção do serviço a ser consultado e da interface, os dados do resultado são mostrados no painel de detalhes e no gráfico.

As opções de visualização, dados textuais e imagem permitem que o usuário decida qual forma de apresentação de dados é mais compreensível ao seu nível de experiência. O critério *Tratamento de Entrada de Informações* (R2) é validado quando se utiliza o painel de seleção. Como exemplo, tem-se a seleção do dia, semana e hora para a busca de novos dados de medição. Contudo, algumas funções ainda apresentam leituras por meio de linhas de comandos. É o caso da configuração da lista de serviço e análise de uma determinada rota. Isso pode dificultar a inserção dos dados e interpretação dos resultados caso o usuário não possua conhecimento técnico suficiente para realizar a referida tarefa. A Figura 4 (c) ilustra a tela em que é solicitada ao usuário a especificação de um caminho para a análise.

A *Utilização de Menus* (R3) também compõe a interface da ferramenta. As funções foram agrupadas no painel de seleção, facilitando a execução das ações que a ferramenta disponibiliza. Da mesma forma se caracteriza o *Controle de Execuções Automáticas* (R4) e o *Controle do Uso de Linguagens Técnicas* (R6), visto que é possível fazer a total intervenção durante a execução da tarefa, num ambiente que dispõe de uma leitura interpretável para qualquer nível de experiência do usuário final.

No que se refere ao *Suporte ao Usuário* (R5), a ferramenta PerfsonarUI apresenta falha. Embora mantenha uma linguagem clara e de fácil compreensão, a ajuda que é

disponibilizada se limita a, somente, informações sobre a versão e links que redirecionam o usuário a uma página externa à aplicação. A Figura 4 (d) ilustra esse cenário.

O ambiente da psUI se diferencia das outras ferramentas desenvolvidas e utilizadas pela comunidade acadêmica, pelo fato de possuir um ambiente que se aproxima do que defende a recomendação *Padronização de Ambientes* (R7). Dispõe de funções agrupadas, área de manipulação e de intervenção e aspectos positivos de leitura, as quais podem ser eficientemente interpretadas por usuários, independente do seu nível de experiência, seja ele avançado, intermediário ou iniciante.

4.2.2. Notas de conformidade das recomendações e grau de adequação obtido

A Tabela 2 apresenta as notas atribuídas a cada recomendação durante a avaliação realizada com a interface da ferramenta psUI. Os dados revelam que a interface da ferramenta psUI está em conformidade com a maioria das recomendações propostas, tendo assim, uma interface com requisitos de usabilidade adequados para usuários não especialistas.

Tabela 2. Notas de conformidade atribuídas a cada recomendação durante a avaliação da interface pSUI.

Recomendação	Nota de avaliação	Justificativa
R1	9	Prioriza a utilização de imagens e gráficos para representação dos dados.
R2	9	Realiza o controle das entradas de dados na aplicação.
R3	8	Agrupa as funções a fim de propor ações mínimas para execução das tarefas.
R4	8	A interface verifica a eficiência e efetividade das intervenções feitas pelo usuário.
R5	4	A interface despreza o conhecimento do usuário ao emitir mensagens de erro técnicas e sem permitir intervenção do usuário como tentativa de solução do problema.
R6	8	A aplicação controla o nível de linguagem técnica utilizada em sua interface.
R7	9	Há uma padronização do ambiente em relação à indicação do usuário não especialista como interface satisfatória para o seu nível de conhecimento técnico.

Após aplicar a Equação 1, foi possível observar que o nível de adequação da psUI foi de 76,5% para o perfil de usuário não especialista avaliado neste trabalho. Esse percentual certamente deveu-se ao fato da interface da ferramenta ter sido desenvolvida com o intuito de oferecer um ambiente organizado e eficiente para a visualização dos dados de medição da rede. Consequentemente, tais características tornam a interpretação desses dados mais rápida para usuários leigos. O percentual não atingiu a totalidade por alguns motivos observados na avaliação da interface descrita na Etapa 2. Por exemplo, a ferramenta não atende na plenitude a recomendação *R5 – Suporte ao Usuário* e parcialmente a recomendação *R2 – Tratamento de Entrada de Informações*.

4.3. Avaliação das ferramentas do CAIDA

O mesmo procedimento adotado na avaliação das interfaces do psUI foi adotado na análise de interfaces de um grupo de oito ferramentas do CAIDA (*Center for Applied Internet Data Analysis*). O CAIDA consiste numa colaboração resultante da parceria entre instituições do governo, pesquisa e entidades comerciais que visam promover melhorias da infraestrutura global da Internet. Entre outras atividades, o CAIDA divulga e mantém uma extensa lista de ferramentas utilizadas no monitoramento da Internet.

As ferramentas avaliadas através das recomendações foram: Corsaro, Iatmon, DNSSTAT, Autofocus, Coralreef, Beluga, Otter e Walrus. O critério de escolha para tais ferramentas se baseou nas características apresentadas e a viabilidade de utilizá-las durante o estudo, uma vez que algumas apresentaram problemas de mal funcionamento, carência de documentação adequada e dificuldade de instalação. Maiores informações sobre tais ferramentas podem ser encontradas no site do CAIDA.

O gráfico apresentado na Figura 5 (a) apresenta o resultado da avaliação das interfaces de tais ferramentas. Como é possível observar, a ferramenta Dnsstat foi a que apresentou nenhuma adequação aos usuários não especialistas, ao passo que a ferramenta Walrus e Otter foram as mais adequadas. O gráfico da Figura 5 (b) apresenta a média geral das notas de conformidade de cada recomendação atribuídas às interfaces avaliadas. Observa-se que as médias estão com valores próximos, demonstrando que todas as recomendações foram utilizadas nas análises das interfaces. A figura também apresenta o desvio padrão de cada média. Para algumas recomendações o desvio padrão foi mais alto por conta de algumas notas de conformidade que teve o valor zero atribuído pelo fato da interface não possuir as características ergonômicas indicadas. Por outro lado, existiram recomendações (ex. R6) que tiveram notas atribuídas em todas as ferramentas, levando a um desvio padrão menor.

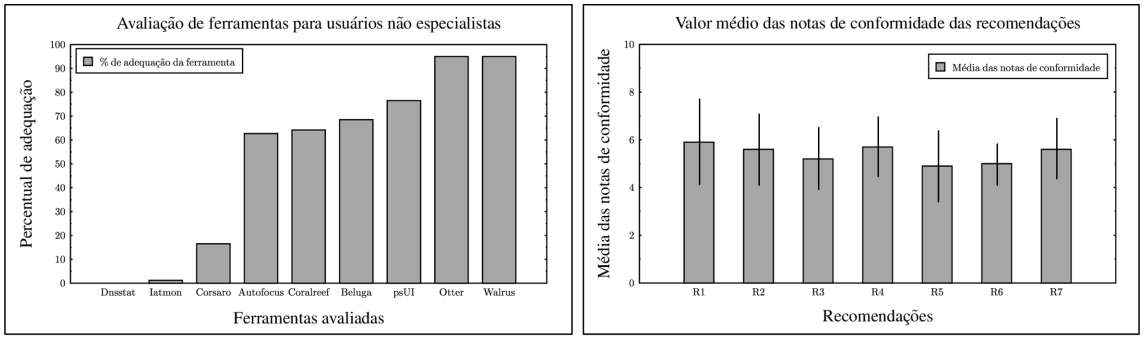


Figura 5. (a) Nível de adequação das ferramentas do CAIDA avaliadas. (b) Média das notas de conformidade das recomendações atribuídas às ferramentas.

5. Conclusões e trabalho futuros

Este artigo descreve uma proposta de recomendações de características ergonômicas para interfaces de sistemas de monitoramento a fim de beneficiar usuários não especialistas. As recomendações propostas podem ser utilizadas tanto na avaliação e quanto no desenvolvimento de sistemas. Além disso, foi também proposta uma metodologia de avaliação das interfaces que, através das recomendações, apresenta o nível de adequação das interfaces para os usuários finais. A avaliação feita com um conjunto de ferramentas do CAIDA, em conjunto com a ferramenta PerfSONAR UI, demonstrou a aplicabilidade das recomendações propostas. Foi possível observar que, de fato, grande parte das ferramentas desenvolvidas para usuários especialistas não são adequadas a usuários leigos.

Como trabalhos futuros, serão realizadas avaliações de ferramentas de monitoramento para usuários de outros perfis. Para isso, será necessário fazer uma nova pesquisa de campo de modo que sejam identificados novos conjuntos de pesos a partir de cada perfil de usuário investigado.

Agradecimentos

Os autores agradecem o apoio dos profissionais da UNIFACS e IFBA por participarem da pesquisa de campo realizada.

Referências

Bajpai, V. and Schönwälder, J. (2015). A survey on internet performance measurement platforms and related standardization efforts. *IEEE Communications Surveys Tutorials*, 17(3):1313–1341.

Bastien, J. M. C. and Scapin, D. L. (1993). Preliminary findings on the effectiveness of ergonomic criteria for the evaluation of human-computer interfaces. In *Human-Computer Interaction, INTERACT '93*, pages 187–188.

Guimarães, V. T., Freitas, C. M. D. S., Sadre, R., Tarouco, L. M. R., and Granville, L. Z. (2016). A survey on information visualization for network and service management. *IEEE Communications Surveys Tutorials*, 18(1):285–323.

Hanemann, A. et al. (2005). PerfSONAR: A service oriented architecture for multi-domain network monitoring. In *ICSOC 2005*, volume 3826 of *LNCS*, pages 241–254. Springer Berlin / Heidelberg.

Hanemann, A. et al. (2006). Complementary Visualization of perfSONAR Network Performance Measurements. In *Internet Surveillance and Protection, 2006. ICISP '06. International Conference on*, pages 6–6.

Jeliazkova, N., Iliev, L., and Jeliazkov, V. (2006). PerfsonarUI - a Standalone Graphical User Interface for Querying perfSONAR Services. In *Modern Computing, 2006. JVA '06. IEEE John Vincent Atanasoff 2006 International Symposium on*, pages 77–81.

Legrand, I., Voicu, R., Cirstoiu, C., Grigoras, C., Betev, L., and Costan, A. (2009). Monitoring and Control of Large Systems with MonALISA. *Queue*, 7(6):40–49.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Oberheide, J., Goff, M., and Karir, M. (2006). Flamingo: Visualizing Internet Traffic. In *NOMS*, pages 150–161.

Rosado, W. E. (2011). Recomendações para a construção de interfaces de sistemas de monitoramento baseadas em técnicas de avaliação de usabilidade. Dissertação de Mestrado - Universidade Salvador (UNIFACS).

Rubin, J. (1994). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. John Wiley & Sons, Inc., New York, NY, USA.

Saaty, T. L. (2008). Relative measurement and its generalization in decision making: Why pairwise comparisons are central in mathematics for the measurement of intangible factors – the analytic hierarchy/network process. *Revista de la Real Academia de Ciencias Exactas, Fsicas y Naturales*, 102(2):251–318.

Swany, M. and Calyam, P. (2014). Second NSF workshop on perfSONAR-based multi-domain network performance measurement and monitoring. Technical report, NSF.

Trilha Principal do SBRC 2016
Sessão Técnica 24
Redes Móveis

CI-PMIPv6 : Uma Abordagem para Handover Interdomínio em Redes Móveis

Nivia Cruz Quental, Paulo André da S. Gonçalves

Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE)
50.740-540 – Recife – PE – Brasil

{ncq, pasg}@cin.ufpe.br

Abstract. *Interdomain handover is a challenging topic in mobile networks. One of the main challenges is to bring together low signalling cost and decentralized mobility management. This paper proposes CI-PMIPv6 (Clustered Interdomain Proxy MIPv6) for interdomain handover. It organizes entities from different domains in a cluster. Information from mobile nodes is difused with minimum additional signaling overhead to PMIPv6. Evaluation results show that, in accordance with the studied scenarios and in comparison to other approaches, CI-PMIPv6 has a minimum signaling cost reduction of 20%. The latency is at least 16% smaller and the packet loss is at least 16% smaller. Further, CI-PMIPv6 obtains greater goodput than other approaches.*

Resumo. *O handover interdomínio é um tópico de pesquisa desafiador em redes móveis. Um dos principais desafios é aliar baixo custo de sinalização com descentralização do gerenciamento de mobilidade. Este artigo propõe o protocolo CI-PMIPv6 (Clustered Interdomain Proxy MIPv6), que organiza entidades de vários domínios em um cluster, difundindo dados dos nós móveis com adição mínima de overhead de sinalização ao PMIPv6. Os resultados de avaliação mostram que de acordo com os cenários estudados e comparado a outras soluções na literatura, o CI-PMIPv6 obteve custo de sinalização ao menos 20% menor, latência no mínimo 16% inferior e perda de pacotes ao menos 16% menor. Ademais, o CI-PMIPv6 obtém goodput superior às demais abordagens.*

1. Introdução

A continuidade de sessões de comunicação entre sistemas finais, denominados nós, sobre redes móveis heterogêneas mediante *handover* tem sido objeto de estudo de pesquisas recentes [Chen et al. 2005] [Zekri et al. 2012]. A ampla difusão de aplicativos para dispositivos móveis, permitindo a interação entre usuários que se deslocam em uma rede móvel, tornam esse problema ainda mais relevante e atual.

No âmbito da mobilidade IP, alguns dos principais desafios envolvem aliar interoperabilidade entre tecnologias de acesso e baixa latência, enquanto permitem a transição entre redes de domínios diferentes de forma transparente. O protocolo Mobile IP (MIP) conserva o endereço IP de um nó móvel fazendo o mesmo trocar mensagens de sinalização com entidades de diferentes domínios à medida que se move para além de sua rede de origem [Johnson et al. 2004]. Essa característica faz do MIP um protocolo com suporte a

mobilidade interdomínio. Porém, o MIP possui como desvantagens o custo de se introduzir processamento nos dispositivos móveis, a necessidade de engajamento de fabricantes de *gadgets* na implementação do protocolo e o *overhead* de comunicação com entidades no núcleo da rede. A extensão Proxy MIP versão 6 (PMIPv6) introduz entidades que monitoram a localização do nó móvel dentro de um domínio local [Gundavelli et al. 2008]. Isso elimina a necessidade de sinalização e processamento diretamente no nó móvel. Assim, o PMIPv6 é um protocolo mais abrangente para os diferentes tipos de aparelhos disponíveis no mercado e mais econômico em termos de consumo de energia. Entretanto, o PMIPv6 acaba por perder o caráter interdomínio do MIP, uma vez que o gerenciamento é limitado a um domínio local. Estudos recentes vêm propondo extensões interdomínio ao PMIPv6 e derivados [Joe and Lee 2012] [Zhong et al. 2010] [Zhou et al. 2010]. Algumas das propostas procuram fornecer esse suporte sem a necessidade de estruturas hierárquicas. Contudo, as soluções propostas na literatura ainda não conseguem aliar baixo custo de sinalização e baixa latência com a compatibilidade com sistemas legados.

Este artigo propõe o protocolo CI-PMIPv6 (*Clustered Interdomain Proxy Mobile* IPv6) como solução interdomínio e intradomínio que atende aos pontos anteriormente citados. O CI-PMIPv6 organiza entidades-âncora de vários domínios em um *cluster*, onde são distribuídas informações sobre os nós móveis com o mínimo de *overhead* de sinalização adicional ao *handover*. O CI-PMIPv6 garante *handover* interdomínio e intradomínio, além de conservar a simplicidade do PMIPv6 com a ampla cobertura do MIP. Adicionalmente, o CI-PMIPv6 não adiciona entidades, não introduz túneis extras e permanece compatível com sistemas legados.

O presente artigo está organizado da seguinte forma: a Seção 2 descreve os conceitos relacionados a gerenciamento de mobilidade IP, *handover* interdomínio e as propostas existentes na literatura. A Seção 3 apresenta o CI-PMIPv6 e suas principais características. A Seção 4 apresenta a metodologia de avaliação de desempenho bem como os resultados obtidos. Por fim, a Seção 5 apresenta as conclusões do trabalho.

2. Mobilidade IP Interdomínio e Trabalhos Relacionados

O protocolo PMIPv6 (*Proxy Mobile IPv6*) conta com a existência de dois tipos de entidades de rede : o *Mobile Access Gateway* (MAG) e o *Local Mobility Anchor* (LMA) [Gundavelli et al. 2008]. A Figura 1(a) ilustra essa arquitetura. O MAG é responsável por enviar atualizações de endereço e manter o controle da localização do nó móvel, evitando a troca de mensagens pelo dispositivo sem fio típica do MIP. O LMA possui função semelhante à entidade *Home Agent* do MIP e gerencia todos os MAGs de sua região.

A Figura 1(b) apresenta o fluxo de mensagens para o PMIPv6 durante um *handover* intradomínio entre dois MAGs. Ao se mover de uma rede para outra, o nó móvel requisita uma nova rota ao novo MAG (NMAG) pela mensagem `Rtr Sol` (*Router Solicitation*) do procolo ICMP. Em seguida, o NMAG solicita ao LMA que atualize suas informações por meio da mensagem *Proxy Binding Update* (PBU), recebendo a resposta de reconhecimento *Proxy Binding Acknowledgment* (PBA). O túnel entre o MAG prévio (PMAG) e o LMA é desfeito e um túnel é estabelecido entre o LMA e o NMAG e assim o *handover* é concluído. Assim, o NMAG pode contactar o nó por meio da mensagem ICMP `RA` (*Router Advertisement*). A extensão *Fast Handovers for Proxy Mobile IPv6* (FPMIPv6) [Yokota et al. 2011] adiciona um esquema de *buffering* e criação de um túnel

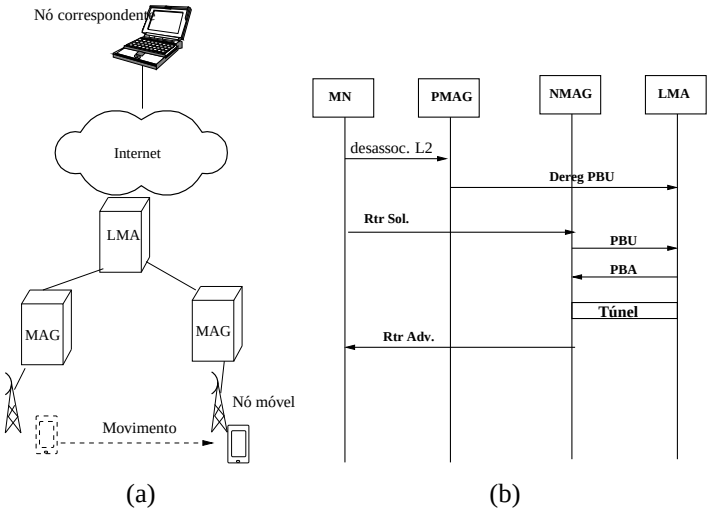


Figura 1. (a) Arquitetura do PMIPv6. (b) Fluxo de mensagens do PMIPv6.

entre o PMAG e o NMAG enquanto as mensagens de controle do *handover* ainda estão sendo trocadas. Essa extensão permite reduzir o número de pacotes perdidos durante o processo. Outras propostas preveem o suporte a *multihoming* [Liebsch et al. 2011], manutenção de rotas otimizadas [Rasem et al. 2012] e otimização do *handover* intra-domínio baseada na formação de subdomínios [Jabir et al. 2014].

Diferentemente do MIP, no PMIPv6 e em suas extensões, os LMAs não possuem conhecimento global da rede, tampouco de domínios externos, como um agente do MIP teria. No MIP, a rede de origem alcança domínios além do seu graças à comunicação com o nó móvel, que se encarrega de informar sobre atualização do seu endereço na rede estrangeira. Como essa comunicação é eliminada no PMIPv6, não é possível rastrear o nó móvel quando este deixa o domínio. Desta forma, é necessária uma abordagem que viabilize o *handover* interdomínio de baixo custo e latência mantendo as vantagens do PMIPv6.

Park *et al.* [S. Park et al. 2010] propõem uma abordagem para prover *handover* interdomínio sem qualquer tipo de otimização ou entidades adicionais, acrescentando repasse das mensagens já conhecidas do padrão a entidades de domínios vizinhos. A Figura 2(a) apresenta o fluxo de sinalização de *handover* interdomínio. Cada domínio possui o seu próprio serviço de autenticação, autorização e auditoria (*Authentication, Authorization, and Accounting* - AAA). Deve haver um túnel extra entre o LMA anterior (PLMA) e o novo LMA (NLMA) além do túnel já existente entre NLMA e NMAG. Assim, é esperado um *overhead* de sinalização dobrado devido a esse repasse. Também é esperado um *overhead* adicional de encapsulamento de cabeçalhos devido ao túnel extra criado entre os LMAs. Uma consequência direta é o aumento da latência. Uma abordagem semelhante é apresentada no trabalho de Zhou *et al.*[Zhou et al. 2010].

Neumann *et al.* [Neumann et al. 2009] propõem o I-PMIP, uma abordagem para *handover* interdomínio que mantém o mesmo LMA até o fim da sessão. Para tal, parte-se da premissa de que os LMAs de domínios vizinhos são fisicamente próximos e que conhecem seus endereços. O LMA de origem mantido como âncora do *handover* é denominado *Session Mobility Anchor* (SMA). O I-PMIP prevê uma entidade adicional chamada *Vir-*

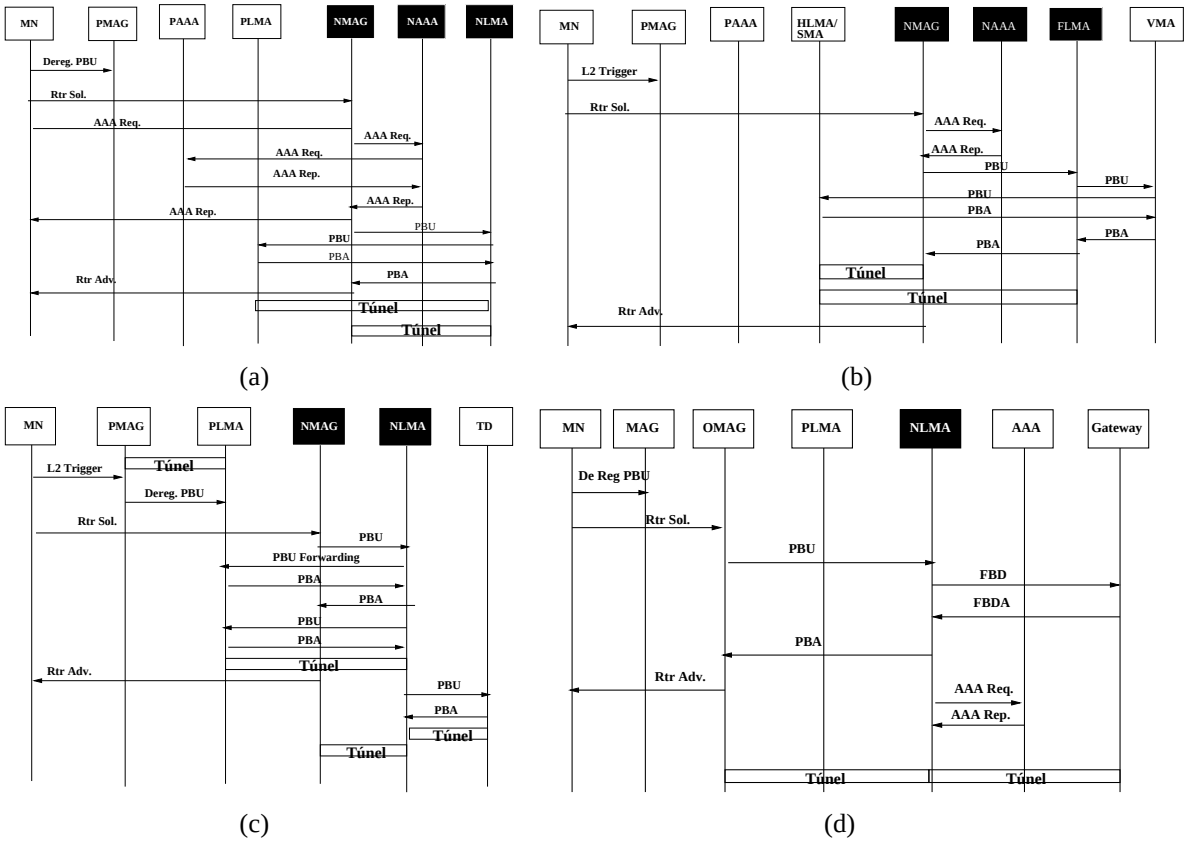


Figura 2. *Handover* interdomínio nas abordagens de: (a) Park *et al.* (b) I-PMIP. (c) EIMHP. (d) No-Gap.

tual Mobility Anchor (VMA), responsável por manter a informação do domínio onde o nó se encontra. A Figura 2(b) apresenta as mensagens de sinalização do I-PMIP no *handover* interdomínio. O SMA repassa os dados ao futuro LMA no novo domínio, que cria um túnel para o novo MAG. O VMA é atualizado cada vez que o nó se move para outro domínio, tornando-se um gargalo de rede e ponto único de falha na arquitetura. Outra desvantagem do I-PMIP é que, assim como na abordagem sem otimização, requer a adição de um túnel além daquele previsto no PMIPv6, aumentando o *overhead* da entrega de pacotes.

Zhong *et al.* [Zhong et al. 2010] propõem o *Enabling Inter PMIPv6 Domain Handover* (EIPMH). O EIPMH introduz uma entidade denominada *Traffic Distributor* (TD), que redireciona dados para o LMA enquanto o nó estiver além do seu domínio original. Os TDs são configurados estaticamente e possuem conhecimento de outros TDs, seus prefixos IP e mapeamento para os LMAs. O TD assume a responsabilidade do LMA de atribuir prefixos IP ao nó móvel além de agir como *gateway* para a rede externa. A Figura 2(c) apresenta a sinalização no EIPMH para o *handover* interdomínio. Assim que o nó se registra no NMAG, o seu NLMA correspondente faz uma consulta ao PLMA de origem por meio da mensagem `PBU_Forwading`. Essa consulta busca informações adicionais do nó e do TD responsável pela comunicação com a Internet. O TD cria um túnel com o novo LMA. Também é criado um túnel entre LMAs além do túnel já existente entre o novo LMA e NMAG. Observa-se que o problema de *handover* interdomínio é apenas

adiado para o nível dos TDs, tornando a solução incompleta. Adicionalmente, o EIPMH introduz dois túneis extras ao PMIPv6, aumentando o *overhead* de entrega de pacotes.

No trabalho de Joe e Lee [Joe and Lee 2012] uma abordagem denominada *no-gap* é apresentada. Nessa abordagem, um tipo especial de MAG é considerado, o *Overlapping* MAG (OMAG), que deve ficar localizado em áreas de sobreposição entre dois domínios. Assim que o nó entra em sua área de cobertura, o OMAG cria dois caminhos simultâneos, um para cada domínio. Assim, o nó recebe informação redundante de ambos LMAs. A Figura 2(d) apresenta o fluxo de chamadas de sinalização para a solução *no-gap*. Além das mensagens PMIPv6, as mensagens FBD e FBDA são trocadas entre o NLMA e um *gateway* para confirmar e obter informação adicional sobre o nó. Adicionalmente o NLMA faz a autenticação do nó. É também necessário criar um túnel entre o *gateway* e o NLMA e entre o NLMA e o OMAG, aumentando o *overhead* de entrega de pacotes. Além disso, a solução *No-Gap* requer uma mudança estrutural no sistema legado, incluindo configuração de roteadores de borda de modo a viabilizar a existência da entidade OMAG.

3. Clustered Interdomain PMIPv6 (CI-PMIPv6)

O CI-PMIPv6 é uma abordagem para *handover* interdomínio e intradomínio que propõe organizar os LMAs de cada domínio em uma estrutura denominada *cluster*, conforme ilustra a Figura 3. Ela permite que MAGs de domínios diferentes se localizem por meio de seus LMAs, previamente conectados ao *cluster*. O gerenciamento do *cluster* é baseado na aplicação Kademlia [Maymounkov and Mazières 2002], uma *Distributed Hash Table* (DHT) tolerante a falhas com performance logarítmica. A Kademlia armazena pares <chave, valor> em pares cujos identificadores (denominados *nodeIDs*) estiverem mais próximos à chave correspondente usando a operação XOR para medição da distância. A escolha por essa DHT advém da sua distribuição eficiente e busca com complexidade O(log(n)), trazendo escalabilidade no caso de haver uma maior quantidade de pares no *cluster*. O *cluster* é configurado estaticamente, dependendo de acordos entre as operadoras de banda-larga. É razoável considerar que os LMAs que farão parte do *cluster* sejam configurados em tempo de implantação do sistema CPMIPv6, pois não se supõe a saída de um LMA, uma entidade fixa, durante a realização de chamadas. A configuração estática é uma premissa que simplifica questões de segurança e manutenção de consistência de dados na atualização das chaves. Desta forma, os serviços de autenticação dentro do domínio podem permanecer inalterados, como no PMIPv6. Cada LMA possui informação de todos os MAGs conectados aos LMAs da vizinhança, graças às operações de armazenamento da Kademlia durante o registro inicial do nó na rede.

O CI-PMIPv6 considera as seguintes premissas: Os MAGs são fisicamente alcançáveis a partir de um LMA de um domínio vizinho; os LMAs funcionam como *gateways* para a Internet; cada serviço de AAA (Autenticação, Autorização e Auditoria) está localizado na mesma rede que o LMA no domínio correspondente; os MAGs não realizam *buffering* adicional em comparação ao que já é feito no PMIPv6. O *cluster* do CI-PMIPv6 distribui pares <chave, valor> entre os LMA onde:

- A chave é o prefixo de endereço do nó móvel na rede de origem;
- O valor é uma estrutura de dados que contém o endereço atual do MAG, do LMA, um identificador do nó móvel na rede de origem, um identificador para o link entre

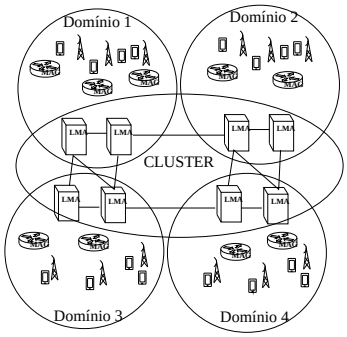


Figura 3. Domínios no CI-PMIPv6.

o nó móvel e a rede de origem e o principal prefixo de endereço fornecido ao nó móvel pela sua rede de origem;
- O *nodeID* de um LMA no *cluster* é o seu endereço IP;
- As chaves e *nodeIDs* do *cluster* estão no espaço de endereços de 128 bits, como todo endereço IPv6 ao invés do endereço de 160 bits da implementação padrão da Kademlia.

Assim, os pares <chave, valor> são armazenados nos LMAs cujos IPs são mais próximos ao prefixo do nó móvel. Essa proximidade é determinada pela operação de XOR, ou seja, não se trata de proximidade geográfica. Entretanto, os IPs com prefixos mais próximos tendem a pertencer a mesma operadora, de modo que seja mais provável o estabelecimento de rotas otimizadas entre eles. A troca de mensagens no *cluster* é feita por meio das primitivas `STORE`, `FIND_NODE` e `FIND_VALUE` da Kademlia. O CI-PMIPv6 introduz também as primitivas `UPDATE` and `DELETE` de modo a atualizar e remover valores no *cluster*.

A Figura 4(a) ilustra o fluxo de mensagens de sinalização no momento do registro de um nó móvel na rede. Ele envia a mensagem ICMP `Rtr Sol` solicitando ao MAG mais próximo uma rota de acesso à rede externa. O MAG primeiramente autentica o nó com o serviço de AAA de seu domínio. Uma vez que o nó é autenticado, o MAG envia uma mensagem de `PBU` ao seu LMA. Até esse momento, o fluxo de registro é idêntico ao do PMIPv6. A partir de então, o LMA envia o comando `STORE` derivado do Kademlia para armazenar essa associação no *cluster*. Assim, os LMAs em domínios vizinhos podem ter acesso ao mapeamento do nó ao seu MAG. A execução do comando `STORE` é assíncrona e não requer aguardar a resposta do *cluster* para o envio da mensagem de `PBA` para o MAG. O nó é registrado e o MAG pode finalmente anunciar a sua rota por meio da mensagem ICPM `RA` para o nó.

A Figura 4(b) apresenta o processo de desregistro. O processo é inicialmente idêntico ao que ocorre no PMIPv6. Após detectar o evento de desligamento, o MAG envia a mensagem `PBU` ao LMA para o desregistro. O LMA aguarda um intervalo de tempo fixo denominado `INITIAL_BINDACK_TIMEOUT` [Johnson et al. 2004] para remover de fato a associação entre nó e MAG. A seguir, o LMA precisa enviar a mensagem `DELETE` ao *cluster*. Essa operação, assim como `STORE`, é assíncrona e independe da sinalização trocada entre LMA e MAG. Assim, o LMA pode enviar a mensagem `PBA` para o MAG e finalizar o processo de desregistro.

A Figura 4(c) apresenta o fluxo de sinalização para o *handover* intradomínio. O

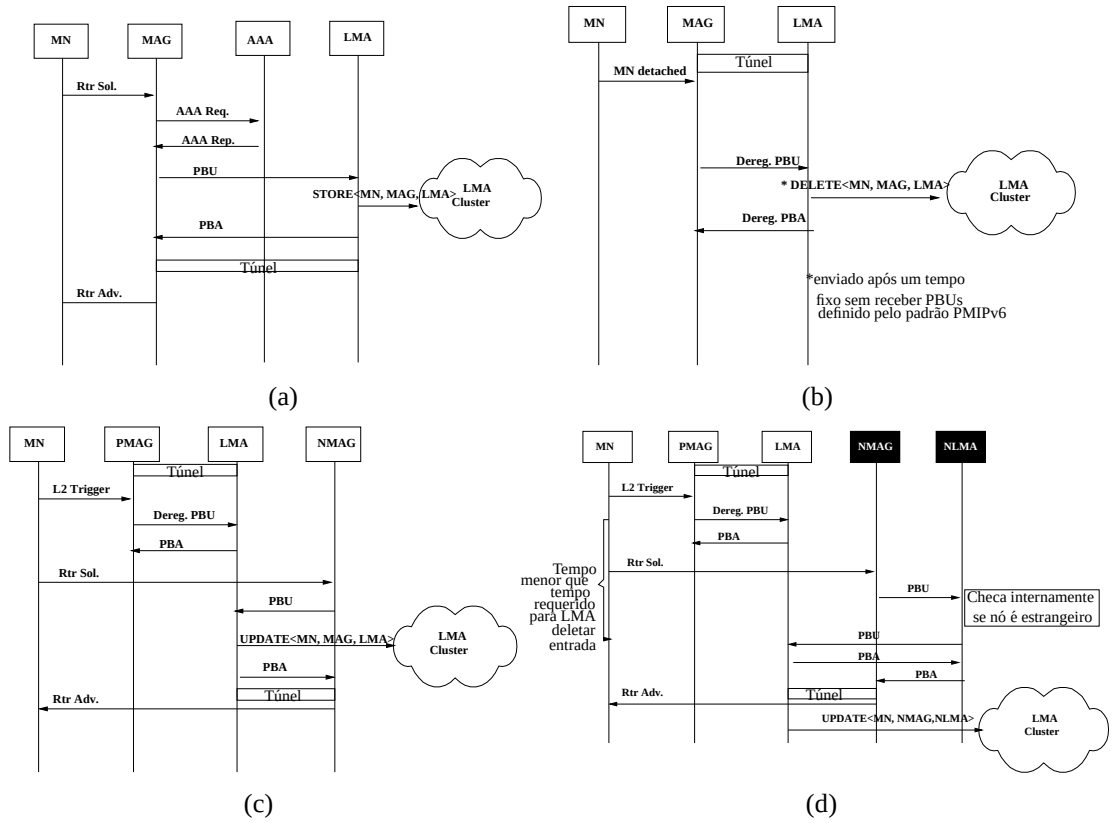


Figura 4. Fluxo de sinalização no CI-PMIPv6: (a) Registro. (b) Desregistro. (c) Handover intradomínio. (d) Handover interdomínio.

fluxo é similar àquele proposto pelo padrão PMIPv6, com a adição da mensagem UPDATE para atualizar o *cluster* após a associação com o novo MAG. É importante observar que, uma vez que o CI-PMIPv6 não remove nenhum tratamento de sinalização do PMIPv6, o sistema pode conviver com entidades legadas. No caso de um LMA legado, ele não pertenceria ao *cluster* e não realizaria *handover* interdomínio e sim apenas intradomínio, que é o esperado no PMIPv6.

A Figura 4(d) apresenta a sinalização para o *handover* interdomínio. O nó envia a mensagem PBU para o PMAG para realizar o desregistro. O PLMA aguarda durante o intervalo INITIAL_BINDACK_TIMEOUT para realizar a remoção definitiva do registro. Quando o nó entra no novo domínio e se comunica com o NMAG, ele envia a mensagem PBU para o NLMA. O NLMA encontra em seu registro a antiga associação do nó com o PMAG e o PLMA e o identifica como nó estrangeiro. Assim, o NLMA envia a mensagem PBU para o PLMA informando que o nó está em um novo domínio. Assim, o PLMA desiste de remover o registro e responde com PBA para o NLMA, que assim, pode enviar o PBA para o NMAG. Uma premissa do CI-PMIPv6 é de que essas mensagens são trocadas em um intervalo inferior a INITIAL_BINDACK_TIMEOUT. Finalmente, o PLMA atualiza o *cluster* com a nova associação.

O maior ganho do CI-PMIPv6 vem da antecipação dos LMAs sobre o conhecimento de informações dos nós móveis, a serem usadas no próximo *handover*. As mensagens trocadas no *cluster* não adicionam custo para o handover em curso, sendo

assíncronas. A vantagem do uso de mensagens assíncronas reside no fato do MAG não precisar aguardar a finalização de procedimentos nos pares no *cluster*, que ficam no núcleo da rede e poderiam se tornar gargalo para a latência do handover. Os MAGs assim podem abstrair a existência do *cluster*, para os quais o protocolo PMIPv6 funciona sem modificações. Assim, é esperado que o custo e a latência de *handover* sejam menores do que nas propostas do Estado da Arte.

4. Avaliação de desempenho

A modelagem é baseada em métricas, parâmetros e equações tiradas dos *frameworks* de avaliação de desempenho de protocolos de mobilidade IPv6 propostos e testados por Taghizadeh *et al.* [Taghizadeh et al. 2012], McNair, Akyldiz e Bender*et al.* [McNair et al. 2001] e Makaya e Pierre *et al.* [Makaya and Pierre 2008]. Por não realizar geração de números aleatórios e seus parâmetros serem baseados em valores médios conforme os frameworks adotados, não são esperados intervalos de confiança nos gráficos. A implementação do modelo foi validada por meio de comparações com os gráficos dos referidos trabalhos. Assim, dados os parâmetros e equações utilizadas, os experimentos são reprodutíveis.

Para as avaliações, considera-se o modelo de mobilidade *Fluid-Flow* [Zhang and Pierre 2008]. É empregado na simulação de padrões de mobilidade veiculares, onde se modela o movimento de nós como um fluxo contínuo de fluído. Tem como parâmetros a velocidade média (v) e as áreas de cobertura do domínio(A_D) e da subrede (A_M). A direção do movimento é uniformemente distribuída em um intervalo de 0 a 2π. A taxa de cruzamento de um domínio (μ_D) é equivalente à taxa de *handover* interdomínio. A taxa de *handover* intradomínio é a diferença entre a taxa de cruzamento de uma subrede (μ_M) e (μ_D). As equações a seguir apresentam as taxas de *handover* interdomínio (Ng) e intradomínio (Nl). O parâmetro *Session-to-Mobility Ratio* (SMR) descreve a mobilidade dos nós, relacionando a taxa de chegada de sessões (λ_S) com a taxa de cruzamento de subrede (μ_M). Caso o SMR se aproxime de zero, a mobilidade é considerada alta.

$$\mu_M = \frac{vL_M}{\pi A_M} \quad (1) \qquad Ng = \mu_D = \frac{vL_D}{\pi A_D} \quad (2)$$

$$Nl = \mu_M - \mu_D \quad (3) \qquad SMR = \frac{\lambda_S}{\mu_M} \quad (4)$$

As métricas avaliadas são: custo de sinalização, latência, perda de pacotes e *goodput*. O custo de sinalização é o número de mensagens de sinalização trocadas durante o *handover*, incluindo-se a distância em saltos entre as entidades envolvidas ($H_{(x-y)}$), o meio de transmissão e o custo de processamento (PC_y), que depende do número de nós móveis considerados em uma eventual busca (N^y_{MN}) e de uma constante normalizadora equivalente à alocação de banda (ς). Para cada mensagem trocada entre duas entidades *x* e *y*, a equação do custo de sinalização (C_{x-y}) é conforme Equação 5.Os parâmetros α e β representam os coeficientes de unidades de custo de transmissão em mensagens/salto nos meios cabeado e sem fio, respectivamente. Caso nenhuma das entidades envolvidas seja móvel, β tem valor zero. Caso a mensagem de sinalização não implique busca em repositório, PC_y pode ser considerado zero. O custo total de sinalização é a soma dos custos de todas as mensagens de sinalização necessárias para o *handover*.

$$C_{x-y} = \alpha(H_{(x-y)}) + \beta + PC_y, \ onde \ PC_y = \varsigma \log N^y_{MN} \quad (5)$$

A latência é medida como a duração da operação de *handover*. Trata-se da soma das latências de cada mensagem de sinalização trocadas por duas entidades *x* e *y* (T_{x-y}). A equação da latência para cada mensagem é dada pela Equação 6. A primeira parte da soma é o *overhead* do meio sem fio e é desconsiderada caso a mensagem não envolva entidades móveis. O parâmetro *q* é a probabilidade de falha na camada de enlace. Ele faz a latência aumentar pois reflete em retransmissões de quadros da camada de enlace. Não se considera o *timeout* na camada de enlace nesse modelo. O parâmetro M_{size} é o tamanho médio de uma mensagem e B_{wl} e B_w são as larguras de banda com fio e sem fio, respectivamente. Os parâmetros L_{wl} e L_w são os atrasos de propagação no meio com fio e sem fio. Finalmente, T_q é o atraso médio de fila em cada roteador.

$$T_{x-y} = \frac{1+q}{1-q}\left(\frac{M_{size}}{B_{wl}} + L_{wl}\right) + H_{x-y}\left(\frac{M_{size}}{B_w} + L_w + T_q\right) \tag{6}$$

O custo e a latência total da execução de um *handover* são calculados como uma soma ponderada dos componentes intradomínio e interdomínio:

$$cost = \frac{custo\,local \times Nl + custo\,global \times Ng}{Nl + Ng} \tag{7}$$

$$latency = \frac{latência\,local \times Nl + latência\,global \times Ng}{Nl + Ng} \tag{8}$$

Ao se avaliar o custo em função da velocidade e *beta*, o mesmo é calculado com base no número médio de *handovers* realizados em uma sessão de comunicação:

$$custo\,na\,sessão = custo\,global \times Ng \times duração\,da\,sessão \tag{9}$$

A perda média de pacotes no *handover* é o número médio de pacotes que deixam de ser enviados/recebidos nesse período. É, portanto, o produto entre a latência de *handover* e a taxa de chegada de pacotes (λ_p):

$$PL = T\lambda_p \tag{10}$$

O *goodput* neste modelo relaciona o tráfego útil de dados com o tráfego total durante uma sessão (TOT):

$$Goodput = \frac{TOT - (P_{size} \times PL_{sessão} + TOT \times PD)}{duração\,da\,sessão}, onde \tag{11}$$

$$TOT = duração\,da\,sessão \times \lambda_p \times P_{size} \tag{12}$$

$$PD = \frac{40 \times H_{túnel}}{(40 + P_{size}) \times H_{MN-CN}} \tag{13}$$

O *goodput* depende diretamente da perda de pacotes e do *overhead* da entrega de pacotes (PD). O PD é o custo do encapsulamento de cabeçalhos IP (40 bytes) em túneis ao longo do caminho entre o nó móvel e seu correspondente. Os parâmetros para se determinar o PD são o tamanho do pacote IP (P_{size}) e o tamanho do túnel em saltos ($H_{túnel}$).

O custo total é medido em função do SMR, da velocidade dos nós e do parâmetro *beta* (β). A latência e a perda de pacotes são medidas em função da probabilidade de falha na camada de enlace. O *goodput* é medido em função do SMR e da probabilidade de falha na camada de enlace.

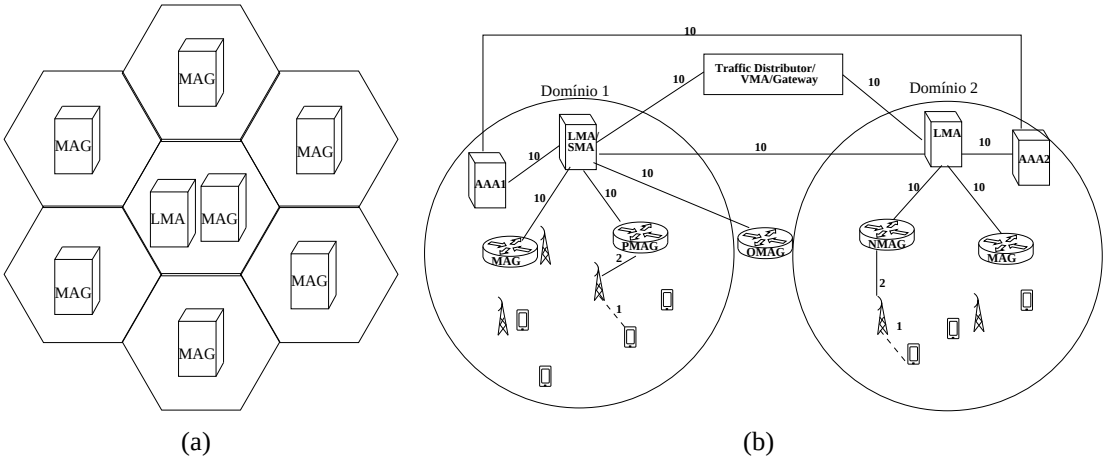


Figura 5. Cenário (a) Domínio com sete subredes. (b) Topologia.

No presente experimento, considera-se o modelo hexagonal de célula, sendo cada célula uma subrede do domínio. Cada subrede é coordenada por um MAG. Considera-se também que o domínio é gerenciado por apenas um LMA. No experimento, um domínio é formado por sete subredes, em um arranjo como o apresentado na Figura 5(a). Cada subrede possui uma área de aproximadamente 1.87 Km^2, ou com 5Km de perímetro. A densidade de concentração dos nós é de 200 nós/Km^2 [Taghizadeh et al. 2012].A Figura 5(b) mostra a topologia de referência usada nos experimentos, incluindo entidades de todas as propostas, inclusive CI-PMIPv6. A distância em saltos entre as entidades é baseada no modelo de Makaya e Pierre [Makaya and Pierre 2008].

Na rede sem fio, considera-se o padrão IEEE 802.11g, com largura de banda nominal de 54Mbps. Na rede cabeada, a largura de banda é de 100 Mbps. O aplicativo *Skype* é tomado como referência para parâmetros de sessão. O tamanho do pacote de dados é de 300 bytes e o tamanho das mensagens de sinalização são de 80 bytes [Chu et al. 2005]. A duração média de uma sessão é de 13 minutos uma vez que o baixo custo de chamadas pelo *Skype* faz com que as chamadas tenham duração maior que aquelas iniciadas por operadoras de telecomunicações [Guha et al. 2006].

O parâmetro *k* de Kademlia usado no CI-PMIPv6 é 10. Esse valor baseia-se em um cenário onde os nós se deslocam a uma velocidade média de 15 m/s (60 Km/h) e, assim, podem cruzar em torno de 10 domínios. Os demais parâmetros são baseados no trabalho de Taghizadeh *et al.* [Taghizadeh et al. 2012] e Chu *et al.* [Chu et al. 2005]: a velocidade varia entre 5 e 40 m/s (18-144 Km/h); a probabilidade de falha no enlace sem fio varia entre 0 e 0,8 (valor de 0,5 para cenários onde o mesmo não varia); o coeficiente α é de 1 mensagens/salto; o coeficiente β é de 10 mensagens/salto; a constante ς tem valor 0,01; o tempo de fila é de 5 ms; o tempo médio de permanência na subrede é de 300 s; o atraso de propagação no meio com fio é de 0,72 μs; o atraso de propagação no meio com fio é de 10 ms; a taxa de chegada de pacotes considerada é de 38 pacotes/s (100 kbps) e a taxa de chegada de sessões considerada é de 0,001 sessões/s.

A Figura 6(a) apresenta os resultados do custo médio de um *handover* em função do fator β. O aumento de β se reflete diretamente no custo, aumentando-o para todos os esquemas. É possível observar que o EIPMH tem o maior custo, uma vez que há

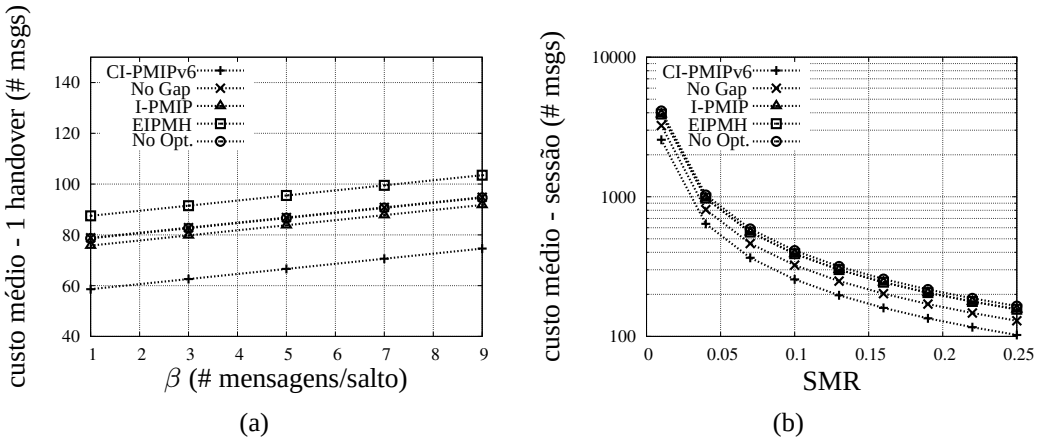


Figura 6. (a) Custo de sinalização x β. (b) Custo x SMR.

mais sinalização devido à existência do TD. O CI-PMIPv6 tem o custo mais baixo devido ao menor número de mensagens trocadas. A presença de um *cluster* que mantenha informações de domínio proativamente no CI-PMIPv6 simplifica a comunicação no momento do *handover* e faz com que menos mensagens sejam necessárias. Os esquemas I-PMIP, No Gap e sem otimização possuem custos similares. O CI-PMIPv6 é no mínimo 21% menos custoso que os esquemas sem otimização I-PMIP, e No-Gap, que tem custos similares.

A Figura 6(b) apresenta a influência do SMR no custo durante uma sessão. O custo tende a se reduzir com o aumento do SMR, pois um alto SMR indica baixa mobilidade. O esquema sem otimização apresenta o pior desempenho e o CI-PMIPv6 apresenta o custo mais baixo, uma vez que requer menos mensagens. O número de mensagens tem um papel importante neste cenário pois com baixos valores de SMR, o número de *handovers* durante uma mesma sessão é maior, aumentando o custo de sinalização. Na situação de mais alta mobilidade, o CI-PMIPv6 tem custo 20% menor que o esquema No-Gap.

A Figura 7 mostra o custo de uma sessão em função da velocidade média dos nós. Altas velocidades levam a uma mobilidade mais intensa e, assim, a um número maior de *handovers*. Mais uma vez, o CI-PMIPv6 apresenta o custo mais baixo, seguido pelo No-Gap. Os esquemas I-PMIP e EIPMH possuem resultados similares e o esquema sem otimização possui o custo mais alto. No caso onde a velocidade é de 40 m/s, o CI-PMIPv6 possui custo ao menos 20% menor que No-Gap.

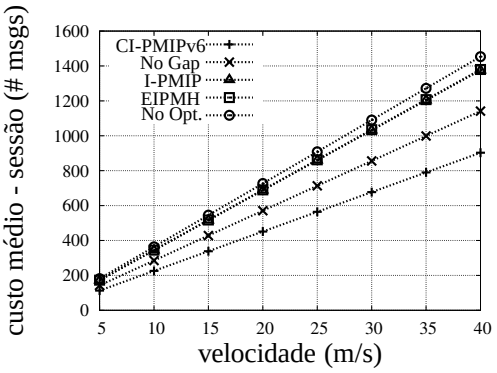


Figura 7. Custo x velocidade.

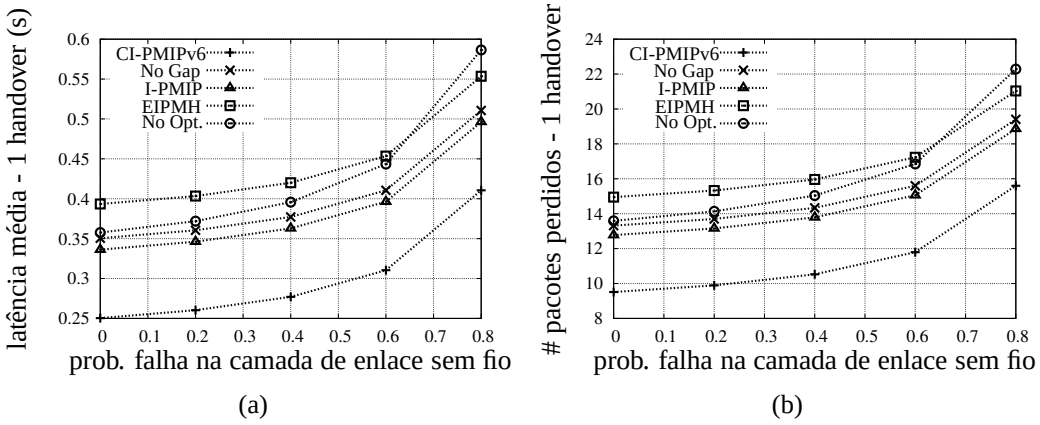


Figura 8. (a) Latência x probabilidade de falha no meio sem fio. (b) Perda de pacotes x probabilidade de falha no meio sem fio.

A Figura 8(a) apresenta a latência média de *handover* quando se varia a probabilidade de falha da camada de enlace sem fio. Essa probabilidade representa a confiabilidade do meio sem fio e pode degradar o desempenho devido a retransmissões. O EIPMH tem a maior latência, influenciada pela maior quantidade de mensagens trocadas até a probabilidade de 0,65. A partir desse ponto o esquema sem otimização mostra uma latência maior. O I-PMIP apresenta resultados ligeiramente melhores que no No-Gap. É importante observar que o CI-PMIPv6 apresenta a menor latência, pelo menos 16% menor no pior caso. Esse melhor desempenho se deve à eliminação de interações desnecessárias no nível da *core network* e na WLAN. A Figura 8(b) apresenta o número de pacotes perdidos no *handover* em função da probabilidade de falha na camada de enlace sem fio. A perda de pacotes está diretamente relacionada à latência do *handover*. Assim, o CI-PMIPv6 tem a menor perda em comparação com as demais propostas, reduzindo a perda em no mínimo 16% com relação à abordagem I-PMIP. A perda pode parecer baixa com valores altos da probabilidade, porém, é importante ressaltar que trata-se de falhas e retransmissões na camada de enlace e desconsidera-se a questão do descarte de conteúdo por *timeout*.

A Figura 9(a) apresenta o *goodput versus* SMR. O *goodput* tende a aumentar e estabilizar enquanto o SMR cresce, pois a mobilidade é reduzida e a perda de pacotes é menor. Já a Figura 9(b) apresenta o *goodput versus* a probabilidade de falha no enlace sem fio. O aumento dessa variável reduz o *goodput* de maneira pouco significativa para todos os esquemas avaliados. Isso ocorre porque a perda de pacotes é baixa devido à baixa participação da componente sem fio no caminho de dados, limitando a influência dessa variável. É importante lembrar que uma probabilidade de 0,8 não significa perda de 80% dos pacotes e sim que há 80% de probabilidade de haver retransmissões na camada de enlace. O CI-PMIPv6 mostra o melhor rendimento considerando os dois cenários. Ele mantem a mesma quantidade de túneis do PMIPv6, evitando o *overhead* de encapsulamento de cabeçalhos IP, enviando assim mais tráfego útil. O EIPMH apresenta o pior *goodput*, pois requer a criação de dois túneis extras além do túnel preexistente.

5. Conclusão

Este artigo apresentou o CI-PMIPv6, uma abordagem interdomínio e intradomínio que gera mudanças mínimas na sinalização do PMIPv6, provê gerenciamento distribuído de mobilidade e antecipa informações de nós móveis para *handovers* futuros por meio de

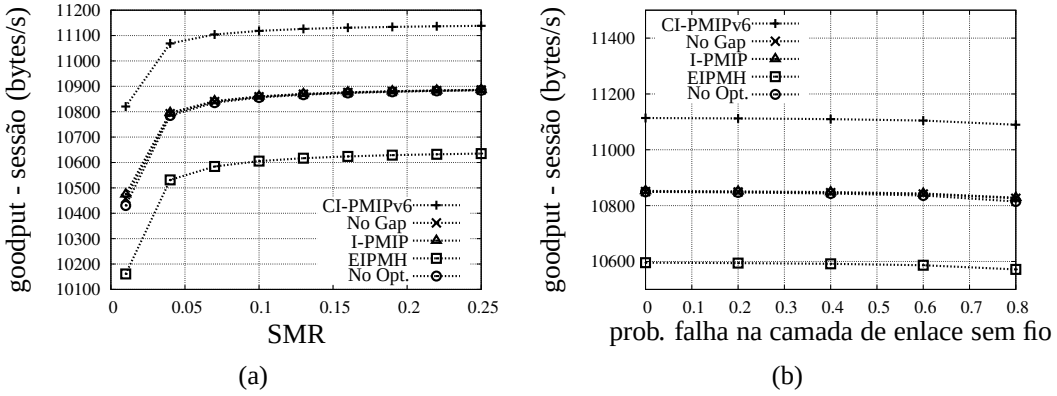


Figura 9. (a) *Goodput* x SMR. (b) *Goodput* x prob. de falha no meio sem fio.

mensagens assíncronas. Além disso, possui as características de não introduzir pontos únicos de falha, não requerer tunelamento extra e não afetar o tratamento de mensagens do protocolo original PMIPv6, sendo compatível com sistemas legados. Como prova de conceito, foi realizada uma modelagem do CI-PMIPv6 e de propostas para handover interdomínio existentes na literatura. Os resultados indicaram que nos cenários estudados, o CI-PMIPv6 reduziu em pelo menos 20% o custo de sinalização, em no mínimo 16% a latência e em no mínimo 16% a perda de pacotes em comparação a outras abordagens da literatura, além de manter um *goodput* superior. Em trabalhos futuros serão feitos mais estudos de simulação em diversos cenários a fim de se quantificar melhor os ganhos do CI-PMIPv6. Também será estudado como prover compatibilidade com o FPMIPv6 e como conservar rotas otimizadas (localized routing) após um handover.

Referências

Chen, L.-J., Sun, T., , and Gerla, M. (2005). USHA: A Practical Vertical Handoff Solution. In *Proc. of the 1st International Conference on Multimedia Services Access Networks*, pages 83–87, Orlando.

Chu, C., Nagarajan, R., Sun, D., and Yang, W. (2005). End-to-End Performance and Reliability Estimation of PacketCable VoIP Services. Telcordia Tech. Report.

Guha, S., Daswani, N., and Jain, R. (2006). An Experimental Study of the Skype Peer-to-Peer VoIP System. Microsoft Research.

Gundavelli, S., Leung, K., Devarapalli, V., Chowdhury, K., and Patil, B. (2008). Proxy Mobile IPv6. RFC 5213.

Jabir, A. J., Shamala, S., Zuriati, Z., and Hamid, N. A. (2014). A Low Cost Route Optimization Scheme for Cluster-Based Proxy MIPv6 Protocol. *Wireless Personal Communications*, 74(2):499–517.

Joe, I. and Lee, H. (2012). An efficient inter-domain handover scheme with minimized latency for PMIPv6. In *Proc. of the International Conference on Computing, Networking and Communications*, pages 332 – 336, Maui.

Johnson, D., Perkins, C., and Arkko, J. (2004). Mobility Support in IPv6. RFC 3775.

Liebsch, M., Muhanna, A., and Blume, O. (2011). Transient Binding for Proxy Mobile IPv6. RFC 6058.

Makaya, C. and Pierre, S. (2008). An Analytical Framework for Performance Evaluation of IPv6-Based Mobility Management Protocols. *IEEE Transactions on Wireless Communications*, 7(3):7.

Maymounkov, P. and Mazières, D. (2002). Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Proc. of the First International Workshop on Peer-to-Peer Systems*, pages 53–65.

McNair, J., Akyildiz, I., and Bender, M. D. (2001). Handoffs for Real-Time Traffic in Mobile IP Version 6 Networks . In *Proc. of the First Global Telecommunications Conference*, pages 3463–3467, San Antonio.

Neumann, N., Lei, J., Fu, X., and Zhang, G. (2009). I-PMIP: An Inter-Domain Mobility Extension for Proxy-Mobile IP. In *Proc. of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, pages 994–999, Leipzig.

Rasem, A., Makaya, C., and St-Hilaire, M. (2012). O-PMIPv6: Efficient Handover with Route Optimization in Proxy Mobile IPv6 Domain. In *Proc. of the IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 47–54, Barcelona.

S. Park, E. Lee, F. Y., Noh, S., and Kim, S.-H. (2010). Inter-domain Roaming Mechanism Transparent to IPv6-Node among PMIPv6 Networks. In *Proc. of the*, pages 1–5, Taipei.

Taghizadeh, A., Wan, T.-C., Budiarto, R., Yap, F. T., and Osman, A. (2012). A performance Evaluation Framework for Network-Based IP Mobility Solutions. *International Journal of Innovative, Computing, Information and Control*, 8(10):7263–7288.

Yokota, H., Chowdhury, K., Koodli, R., Patil, B., and Xia, F. (2011). Fast Handovers for Proxy Mobile IPv6. RFC 5949.

Zekri, M., Jouaber, B., and Zeghlache, D. (2012). A Review on Mobility Management and Vertical Handover Solutions over Heterogeneous Wireless Networks. *Computer Communications*, 35(17):2055–2068.

Zhang, L. J. and Pierre, S. (2008). Performance Analysis of Fast Handover for Hierarchical MIPv6 in Cellular Networks. In *Proceedimgs of the IEEE Vehicular Technology Conference*, pages 2374–2378, Singapore.

Zhong, F., Yang, S., Yeo, C. K., and Lee, B. S. (2010). Enabling Inter-PMIPv6-Domain Handover with Traffic Distributors. In *Proc. of the 7th IEEE Consumer Communications and Networking Conference*, pages 1–5, Las Vegas.

Zhou, D., Zhang, H., Xu, Z., and Zhang, Y. (2010). Evaluation of Fast PMIPv6 and Transient Binding PMIPv6 in Vertical Handover Environment. In *Proc. of the IEEE International Conference on Communications*, pages 1550–3607, Cape Town.

GROUPS-NET: Roteamento Ciente de Encontros de Grupos em Redes Móveis D2D

Ivan Oliveira Nunes[1], **Pedro O. S. Vaz de Melo**[1], **Antonio A. F. Loureiro**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos, 6627 – Pampulha, 31270-901- Belo Horizonte - MG - Brazil

{ivanolive,olmo,loureiro}@dcc.ufmg.br

Abstract. *The increasing ubiquity of mobile devices with inter-connection capability leveraged the possibility of device-to-device communication. In the next generation celular networks, device-to-device communication is already considered a fundamental feature. Outside celular communication context, such kind of mobile Ad-hoc networks are also refered as Delay Tolerant Networks (DTNs). A fundamental problem of DTNs is on how to define forwarding algorithms that achieve, at the same time, high delivery ratio and low network overhead. In the past decade, several algorithms were proposed to achieve those goals. Among them, probabilistic and social aware approaches have remarked themselves as the most succesful strategies. However, existent social aware algorithms do not account for the underneath dynamics and evolving nature of social groups. In this work, we propose a probabilistic opportunistic routing protocol which is aware of group meetings and how these meeting properties evolve over time. In large scale scenarios, our algorithm achieve approximately the same delivery ratio of state-of-art solution with up to 40% less network overhead.*

Resumo. *A crescente ubiquidade de dispositivos móveis com capacidade de interconexão alavancou a possibilidade de comunicação dispositivo-dispositivo. Na próxima geração de comunicação celular, a comunicação dispositivo-dispositivo já é considerada uma funcionalidade fundamental. Fora do contexto de redes celulares, esse tipo de rede Ad-hoc também é referida como rede tolerante a atrasos (do inglês, DTN). Um problema fundamental em DTNs está em como definir algoritmos de encaminhamento de mensagens que, ao mesmo tempo, alcancem alta taxa de entrega aos destinatários e baixa sobrecarga na rede. Nas últimas décadas vários algoritmos foram propostos. Entre eles, algoritmos probabilísticos e algoritmos cientes de contexto social se destacaram como estratégias de maior sucesso. Porém, os algoritmos cientes de contexto social existentes na literatura não consideram a natureza dinâmica e evolutiva das relações de grupos sociais. Neste trabalho, propõe-se um protocolo probabilístico de roteamento oportunístico que considera reuniões de grupos e como as propriedades dessas reuniões evoluem ao longo do tempo. Para testes realizados em cenários de larga escala, o algoritmo proposto apresentou, em relação à solução do estado da arte, aproximadamente a mesma taxa de entrega com redução de 40% na sobrecarga da rede.*

1. Introdução

A crescente demanda por conteúdos populares com grande volume de dados como vídeos, imagens e músicas alavancou a utilização de comunicação dispositivo-dispositivo (D2D)

na próxima geração das redes celulares [Li et al. 2014]. Em redes celulares com suporte a D2D, os dispositivos podem se comunicar diretamente sob um moderado controle das estações bases. Essa nova arquitetura permite que algoritmos de controle centralizados na estação base atuem dinamicamente na política de encaminhamento de dados dos dispositivos. Além disso, muitas aplicações sensíveis ao contexto, que envolvem a descoberta e comunicação com dispositivos próximos, podem se beneficiar da comunicação D2D, economizando em potência de transmissão (já que os dispositivos estarão fisicamente próximos) e aumentando a eficiência espectral da rede celular [Fodor et al. 2012]. Espera-se que a comunicação D2D seja uma funcionalidade chave na próxima geração de comunicação celular [Lei et al. 2012].

Nesse contexto, a utilização de algoritmos bem estabelecidos destinados a Redes Tolerantes a Atraso (DTNs) é vista como uma forma de viabilizar a comunicação D2D em redes celulares. Esses algoritmos de encaminhamento em redes DTN, por sua vez, baseiam-se primordialmente na estrutura estática das comunidades sociais que são formadas entre as pessoas para, assim, definir as suas políticas de encaminhamento. Porém, a detecção distribuída de comunidades é uma tarefa complexa. Como discutido por Hui et al. [Hui et al. 2007], algoritmos existentes para detecção distribuída de comunidades atingem, no máximo, 85% de precisão em relação a suas versões centralizadas. Isso claramente impõe dificuldades e vieses à utilização de algoritmos baseados em comunidades. Além disso, algoritmos baseados em comunidades estáticas não consideram a dinamicidade da mobilidade humana, não levando em conta que essas comunidades podem evoluir e se modificar ao longo do tempo. Por fim, a detecção de comunidades depende da calibração de parâmetros para cada cenário específico, o que torna sua utilização inviável em aplicações de tempo real.

Considerando essas questões, no presente trabalho propõe-se um algoritmo de encaminhamento ciente de contexto social que substitui a utilização de comunidades sociais estáticas pelo conhecimento da dinâmica de encontros de grupos. No contexto deste trabalho, um grupo é definido com um conjunto de pessoas que se encontram de forma recorrente de modo que a maioria dessas pessoas esteja sempre presente nos encontros. Ao contrário das comunidades, do ponto de vista de implementação distribuída, é muito simples identificar encontros de grupos. Basta que cada dispositivo armazene a lista de nós que permaneceram próximos a ele por tempo suficiente para considerar que esteja ocorrendo uma interação social. Dessa forma, propõe-se utilizar a ciência de reuniões de grupos em um esquema de roteamento probabilístico grupo-a-grupo que leva uma mensagem de um usuário origem a um usuário destino. As principais contribuições deste trabalho são:

- Uma metodologia para identificar e rastrear encontros de grupos em *traces* de contatos.
- Um algoritmo de roteamento oportunístico para redes D2D sensível ao histórico de encontros sociais de grupos que elimina a necessidade de detecção distribuída de comunidades.
- Uma análise comparativa que avalia a taxa de entrega e sobrecarga da rede em cenários reais (emulados). O algoritmo proposto revela ser mais apropriado que o atual estado da arte em diversas situações.

As demais seções do artigo estão organizadas como a seguir. A Seção 2 revisa alguns trabalhos relacionados e discute as respectivas contribuições nas áreas de mobilidade humana, detecção de comunidades e roteamento oportunístico. A Seção 3 descreve

as etapas metodológicas para identificar e rastrear grupos a partir de *traces* de contatos. Na Seção 4 é introduzido o algoritmo proposto: GROUPS-NET (*Groups ROUting in Pocket Switched-NETworks*). Na Seção 5 é apresentada a metodologia de testes e a comparação entre os algoritmos GROUPS-NET e Bubble Rap. Finalmente a Seção 6 traz as considerações finais e os trabalhos futuros.

2. Trabalhos Relacionados

Mobilidade Humana. Soluções de roteamento para ambientes móveis não podem ignorar as propriedades que governam a mobilidade humana. Gonzalez et al. [Gonzalez et al. 2008] utilizaram uma grande base de dados de telefonia, *Call Detail Records* (CDR), para caracterizar padrões individuais de mobilidade. Seus resultados mostram que o deslocamento radial nas trajetórias humanas, denominado *Radius of Gyration*, segue uma distribuição de probabilidades *power-law* com corte exponencial. Suas análises também revelaram que as trajetórias humanas apresentam periodicidade. Em relação às dinâmicas de mobilidade em grandes cidades, alguns trabalhos avaliaram o impacto gerado por eventos de larga escala. Calabrese et al. [Calabrese et al. 2010] analisaram mais de 1 milhão de dados de telefonia celular para revelar o impacto desse tipo de evento no trânsito de veículos e na mobilidade. Segundo os autores, a compreensão da origem das pessoas que comparecem a um dado evento é fundamental para propor políticas para mitigação de congestionamentos no tráfego. Silva et al. [Silva et al. 2014] propuseram uma técnica de visualização, denominada *City Image*, que captura as transições típicas entre pontos de interesse (PoIs) em cidades usando dados públicos coletados da rede social *Foursquare*. "*City Images*" podem ser compreendidas como intuições para os padrões de mobilidade de cada cidade. Existem também diversos trabalhos no sentido de caracterizar e predizer condições do trânsito usando diversos tipos de fontes de dados como em [Lu and Cao 2003, Bauza et al. 2010]. Esses estudos investigam, por exemplo, como condições climáticas afetam as condições do tráfego. Nenhum dos trabalhos supracitados, porém, analisou os padrões de mobilidade de grupos.

Detecção de Comunidades. Em uma implementação real, a detecção distribuída de grupos é uma tarefa simples, pois um grupo é definido somente pelas pessoas que estão geograficamente próximas durante um período de tempo. No entanto, a tarefa de detectar grupos de forma centralizada em *traces* de contato de larga escala não é simples. Tal tarefa consiste em analisar milhões de contatos que ocorreram das mais diversas formas e durações. Assim, neste trabalho aplicamos a consolidada ideia de detecção de comunidades em grafos para detectar grupos em traces de mobilidade. Desde sua introdução, algoritmos de detecção de comunidades em grafos tem atraído grande atenção. Algoritmos de detecção de comunidades podem ser classificados de acordo com duas características: com sobreposição X sem sobreposição e estáticos X dinâmicos. Dentre vários algoritmos propostos, [Palla et al. 2005] e [Gregory 2010] se destacaram como dois dos mais populares e efetivos para detecção de comunidades em grafos estáticos. Trabalhos como [Nguyen et al. 2011] focam em propôr adaptações e novos algoritmos voltados especificamente para grafos dinâmicos, considerando questões como eficiência computacional. Existem também esforços que buscam mapear a evolução das comunidades detectadas em diferentes tipos de redes complexas [Palla et al. 2007]. No presente estudo, são utilizadas essas metodologias desenvolvidas, especificamente *Clique Percolation Method* (CPM) [Palla et al. 2005], para detectar reuniões de grupos a partir de *traces* de contatos.

Roteamento Oportunístico. Dentre as soluções propostas de roteamento oportunístico, destaca-se o roteamento probabilístico e o roteamento sensível ao contexto social. O

roteamento probabilístico em DTNs foi primeiramente introduzido por Lindgren et al. [Lindgren et al. 2003]. A ideia central é que encontros entre pares que ocorreram mais recentemente são mais prováveis de ocorrerem no futuro próximo. O algoritmo PROPHET, que implementa essa ideia central, obteve grande sucesso, mas foi superado por algoritmos que consideram as relações sociais entre os usuários da DTN. Nesse sentido, Hui et al. [Hui et al. 2011] usaram a estrutura de comunidades sociais em redes móveis combinada com popularidade de nós para projetar um protocolo de encaminhamento de mensagens denominado *Bubble Rap*. Até a presente data, *Bubble Rap* é o algoritmo que apresenta os melhores resultados em termos de maximização da taxa de entrega de mensagens e minimização da sobrecarga da rede. A aplicação do Bubble Rap a redes celulares D2D é proposta em [Li et al. 2014]. Diferente de [Hui et al. 2011], a solução proposta neste trabalho busca detectar grupos de pessoas que estão de fato reunidas, próximas espacialmente e temporalmente. Em [Hui et al. 2011], os autores constroem um único grafo que agrega todos os contatos entre pares que ocorreram ao longo da duração da base de dados e identificam comunidades de nós que possuem laços sociais mais fortes. Essa abordagem impede de levar em conta as mudanças no comportamento social ao longo do tempo, i.é., o dinamismo social. Além disso, é necessário detectar comunidades de forma distribuída, o que é, ainda, um grande desafio. A metodologia proposta neste trabalho é capaz de capturar a natureza dinâmica dos encontros sociais e não necessita da detecção dinâmica de comunidades. Finalmente, esse trabalho propõe e avalia comparativamente uma nova forma de utilizar a dinâmica dos encontros sociais para projetar um algoritmo de encaminhamento de mensagens em redes D2D.

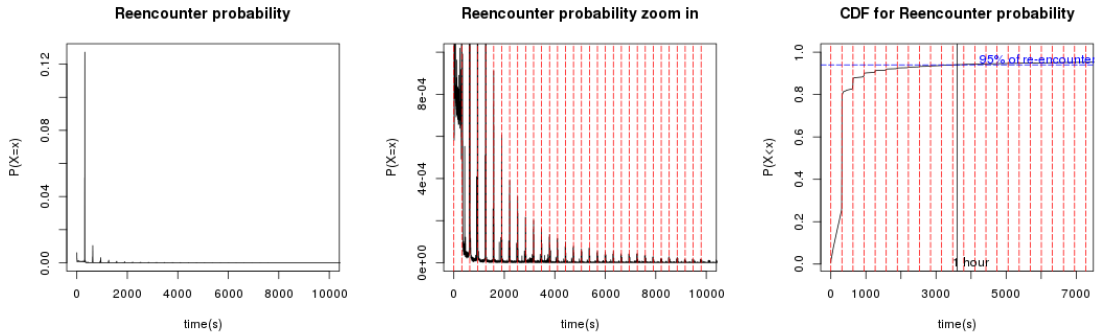
3. A Dinâmica dos Encontros de Grupos

Embora em um cenário real a detecção distribuída de uma reunião de grupo seja muito simples, para permitir a avaliação experimental do algoritmo proposto é preciso detectar reuniões de grupos a partir de *traces* de contatos, que são o tipo de dado tipicamente utilizado para avaliar algoritmos de encaminhamento D2D. Nesta seção são descritos a metodologia utilizada para detectar grupos a partir de *traces* de contatos entre usuários móveis e os principais resultados que motivaram a proposta de um algoritmo de roteamento oportunístico ciente do contexto de grupos.

3.1. Identificação dos Encontros de Grupos

3.1.1. Modelando *Traces* de Contatos com Grafos

Para detectar encontros de grupos propõe-se um modelo de grafo social temporal. Primeiramente, divide-se a base de dados de contatos em janelas de tempo tw (a duração ideal para tw é discutida na Seção 3.1.2). Contatos dentro de uma mesma janela de tempo tw são agregados em um grafo de contatos $Gc(V, E[tw = i])$, onde V é o conjunto de nós da base de dados (i.e., pessoas) e E é o conjunto de arestas que representam contatos de proximidade entre um par de entidades em V. Assim, nesse modelo, o processamento do *trace* de contatos resultará em um conjunto de grafos subsequentes, com arestas ponderadas e não direcionadas: $S = \{Gc(V, E[tw = 0]), Gc(V, E[tw = 1]), ..., Gc(V, E[tw = n])\}$. O peso de cada aresta $(v, w) \in E$ pode ser calculado $i)$ pelo número de contatos entre v e w durante a fatia de tempo tw ou $ii)$ pela soma da duração dos contatos entre v e w, quando essa informação se encontra disponível na base de dados. Neste trabalho foram utilizadas uma base de dados em que a duração dos contatos não se encontra disponível, por isso nesse caso computa-se o peso das arestas de acordo com a opção $i)$. Também foi utilizada uma base de dados onde a duração dos contatos está disponível e nesse tipo de base foi utilizada a opção $ii)$.



(a) (b) (c)

Figura 1. P.D.F. do tempo x (em segundos) até o próximo contato. Linhas pontilhadas vermelhas mostram um intervalo fixo de 318 segundos em que picos de reencontros ocorrem. Isso mostra que, na maioria das vezes, a aquisição dos dados de proximidade é feita em períodos fixos de 318 segundos. A probabilidade de um par de nós se reencontrar tem distribuição aproximadamente exponencial e 95% dos reencontros ocorrem em menos de uma hora.

3.1.2. Caracterização dos Dados

Considerado uma janela de tempo tw pré-definida, por exemplo 30 minutos, um único contato ao longo de toda essa fatia de tempo não necessariamente significa que este dado par de entidades está interagindo. Isso pode ser causado por uma interseção na trajetória dos nós e não deve necessariamente ser considerada uma interação. Por isso é importante definir um limiar para o peso mínimo w_{th} de uma aresta para que essa aresta seja considerada representativa de um encontro social. É visível que para definir ambos, o tamanho da janela de tempo tw e limiar do peso das arestas w_{th}, é necessário avaliar as propriedades do *trace* de contatos. Nesta seção é realizada uma caracterização das bases de dados utilizadas neste estudo a fim de definir corretamente tw e w_{th}. A análise a seguir pode igualmente ser aplicada para outras bases de dados de contatos.

No presente estudo, foram utilizados os traces MIT Reality Mining [Eagle and Pentland 2006] e Dartmouth [Henderson et al. 2008], traces de contatos contendo 80 e 1200 usuários, respectivamente. No trace MIT Reality Mining os usuários monitorados residem em dois prédios da universidade e foram monitorados por diversos meses. Embora o *trace* MIT Reality Mining consista de um cenário específico e de pequena escala, como ele foi o trace utilizado para avaliar o algoritmo do estado da arte Bubble Rap, decidiu-se considerá-lo neste trabalho. Contatos foram registrados quando dois usuários se encontravam a uma distância inferior a 10 metros. Uma entrada no trace de contatos é composta dos IDs do par de nós e da data e hora em que foi registrado o contato do par. Vale mencionar que *traces* de geo-localização (como os obtidos por GPS) podem ser convertidos para *traces* de contatos, definindo uma distância mínima que deva ser considerada um contato entre duas entidades. Por isso, a metodologia aqui apresentada pode também ser aplicada nesse tipo de base de dados. O *trace* Dartmouth monitora todos os usuários de um campus universitário ao longo de dois meses e, além de apresentar os contatos, traz também informação sobre a duração desses contatos. Ao longo dessa seção, devido à limitação de espaço, será apresentada apenas a caracterização para trace MIT Reality Mining, embora a metodologia de caracterização possa ser igualmente aplicada no *trace* Dartmouth e em outros.

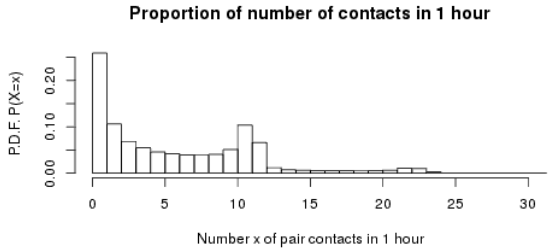
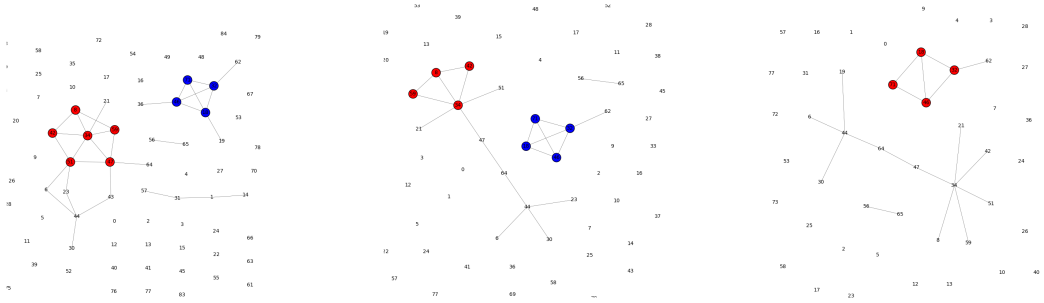


Figura 2. P.D.F. do número de contatos por hora dos pares.



(a) 7AM (b) 8AM (c) 9AM

Figura 3. Detecção de grupos usando o CPM, no *trace* MIT Reality Mining, com $tw = 1h$ em três janelas de tempo consecutivas, em 5 de Fevereiro de 2009. Apenas arestas com $w_{th} >= 2$ foram representadas.

Para permitir a detecção e o rastreamento dos grupos é necessário compreender a base de dados utilizada com o objetivo de evitar erros potenciais, tais como erros de amostragem e de inconsistência nos dados, que são gerados no processo de aquisição. Essa avaliação é realizada com o objetivo de definir i) o tamanho de tw, ou seja, a fatia de tempo ideal para dividir o *trace* e ii) w_{th}, que é o limiar para o número de contatos (ou duração de contatos) que separa contatos sociais e aleatórios.

Primeiramente, analisa-se o tempo entre os re-encontros de pares, i.e, dado que um par se encontrou, qual é o tempo esperado até o próximo encontro? A Figura 1 mostra que o comportamento dos re-encontros é periódico com moda em períodos de cinco minutos (linhas vermelhas pontilhadas). Esse comportamento indica que o sistema implantado para aquisição dos dados atua de cinco em cinco minutos, na maioria das vezes, mas por alguma razão, pode também atuar em intervalos menores. A partir da função de distribuição acumulada C.D.F. (*cumulative distribution function*) da probabilidade de re-encontros 1-c, nota-se que aproximadamente 95% dos re-encontros podem ser capturados com um tw de uma hora. Por essa razão, para o *trace* MIT Reality Mining, considerou-se $tw = 1h$.

Em seguida, analisa-se a proporção do número de contatos dentro do intervalo $tw = 1h$ para que seja possível definir w_{th}. A Figura 2 mostra que 27% dos pares que se encontram em um intervalo de uma hora só se encontram uma vez. Assumimos estes encontros como coincidências nas trajetórias dos pares. Para frequências de encontros entre 2 e 12, o gráfico mostra valores mais uniformes, entre 5 e 10%. Para frequências maiores que 12 encontros, a probabilidade se torna bem baixa, fato que é consistente com

a premissa de que a aquisição dos dados ocorre, na maioria das vezes, em períodos de 5 minutos. De 2 a 12 encontros a função de densidade de probabilidade P.D.F. (probability density function) apresentou valores similares quando comparados com $P(X = 1)$. Por essa razão, para o *trace* MIT Reality Mining com $tw = 1h$, definiu-se $w_{th} = 2$.

Em suma, por meio da caracterização dos dados foi possível definir que dois ou mais contatos no intervalo de uma hora são suficientes para considerar uma interação como social no *trace* MIT Reality Mining. Por meio de análise similar, definiu-se para o trace Dartmouth $tw = 1h$ e $w_{th} = 10$ minutos de duração de contato.

3.1.3. Detecção de Grupos

Após definir valores para tw e w_{th}, define-se um grupo como a seguir:

- ***Definição de grupo:*** *Um grupo é um conjunto de nós altamente conectados em $Gc(V, E[tw = i])$, que é o grafo gerado a partir da i-ésima subdivisão do trace S, após eliminar contatos entre pares sem laços sociais e eliminar arestas cujo peso seja inferior ao limiar w_{th}.*

Até aqui, foi estabelecido um modelo para representar interações sociais que consistem de grafos gerados a partir de *traces* de contatos divididos em janelas de tempo. Seguindo a definição acima, faz-se necessário identificar conjuntos de nós *altamente conectados* em um grafo. Para isso, consideramos a vasta literatura de algoritmos de detecção de comunidades em grafos [Fortunato 2010]. Uma comunidade é comumente definida como um conjunto de nós mais densamente interconectados em grafos . Existem na literatura diversos algoritmos para detecção de comunidades em grafos, como [Xu et al. 2013, Nguyen et al. 2011, Gregory 2010]. Dos algoritmos existentes, neste trabalho é utilizado o *Clique Percolation Method* (CPM) [Palla et al. 2005]. As principais razões para se utilizar o CPM são que os membros de suas comunidades podem ser alcançados por conjuntos de nós fortemente conectados e que pode haver sobreposição entre comunidades. Essa última propriedade é essencial, pois a maioria dos grafos sociais caracteriza-se por ter comunidades sobrepostas e aninhadas [Palla et al. 2007]. Para cada grafo de janela de tempo $Gc(V, E[tw = i])$ computa-se o CPM.

No CPM, uma comunidade é definida como a união de todos os k-cliques (subgrafos completos de tamanho k) que podem alcançar uns aos outros por uma série de k-cliques adjacentes (onde adjacência significa compartilhar $k - 1$ nós). O parâmetro k do CPM limita o tamanho mínimo das comunidades detectadas. Nos últimos anos o CPM consolidou-se como um dos mais bem sucedidos métodos para detecção de comunidades [Peel 2010]. Com o propósito de detectar encontros de grupos, foi atribuído o valor $k = 3$, assim, considera-se grupos de três ou mais pessoas. A figura 3 mostra grupos detectados com o CPM no *trace MIT Reality Mining* em 05 de Fevereiro de 2009, às 7, 8 e 9 horas da manhã; janelas de tempo consecutivas de duração de uma hora.

3.2. Propriedades dos Encontros de Grupos

Depois de aplicar a metodologia para detecção dos encontros de grupos apresentada na Seção 3, aqui serão discutidas algumas das propriedades mais relevantes identificadas que motivaram a aplicação do conhecimento de grupos no roteamento oportunístico em

D2D/DTNs. Uma visão mais detalhada da dinâmica e da evolução dos encontros de grupos pode ser encontrada em nosso trabalho anterior [Nunes et al. 2016].

A Figura 4-a mostra a frequência dos re-encontros de grupos, i.e, dado que um grupo se encontrou pela primeira vez no tempo $t = 0$, como os re-encontros desse grupo estão distribuídos ao longo das horas seguintes? O resultado mostra que a massa de probabilidade se concentra em torno de picos em períodos de 24 horas (representados pelas linhas vermelhas pontilhadas). Isso indica que os encontros de grupos são altamente periódicos considerando periodicidade diária. Também é possível notar que em períodos de 7 dias (representados por linhas pontilhadas verdes) os picos se tornam maiores, evidenciando a presença moderada de periodicidade semanal. Esse resultado faz sentido já que pessoas estão sujeitas horários e rotinas. Esse resultado motivou o experimento seguinte que tenta responder a seguinte pergunta: é possível utilizar encontros de grupos no passado para prever futuros encontros?

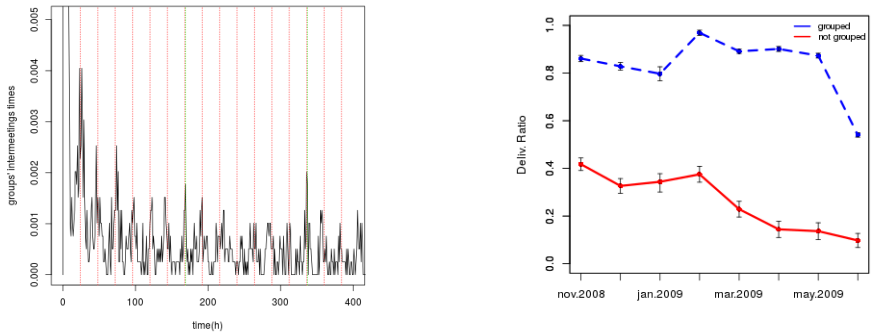
Neste experimento, seleciona-se um dos nós do *trace* como origem de uma mensagem e emula-se a transmissão da mensagem de forma epidêmica, ou seja, toda vez que um nó que já recebeu a mensagem encontra com um nó que ainda não a recebeu, a mensagem é propagada. A propagação da mensagem é emulada tendo cada um dos nós do *trace* como origem. Para avaliar a utilidade da informação sobre os contatos de grupos, os nós são então divididos em duas classes: nós que estiveram em grupo com a origem nos últimos 30 dias e nós que não estiveram em grupo com a origem no mesmo período. Atribuiu-se um tempo máximo de 7 dias para propagação epidêmica das mensagens. Ao fim dos sete dias calcula-se a proporção dos nós de cada uma das duas classes que recebeu a mensagem, ou seja, a taxa de entrega.

Como apresentado na figura 4-b, para difusões iniciadas em diferentes meses (a-h) no *trace* MIT Reality Mining, a taxa de entrega para nós que estiveram em grupo com a origem (90%) é significantemente mais alta do que para nós que não estiveram em grupo com a origem (40%). Este resultado motivou a proposta de um algoritmo que explore grupos detectados para realizar roteamento oportunístico em DTNs. Pode-se notar que o mês de junho apresentou taxas de entrega bem menores em relação aos meses anteriores. Isso ocorreu porque o mês de junho é o início do período de férias de verão, quando muitos estudantes deixam o campus.

4. Roteamento Probabilístico Ciente de Grupos

No algoritmo proposto, procura-se explorar o histórico recente de encontros de grupos para definir a rota grupo-a-grupo mais provável para a propagação da mensagem. O algoritmo tem como entrada o conjunto de grupos identificados no período T (e.g. 24h, 7 dias etc) do passado recente que deve ser considerado para estimar as rotas mais prováveis. Além disso, o parâmetro NRP indica o número de caminhos redundantes a considerar para o encaminhamento da mensagem.

Para cada grupo identificado no período de T horas antes do inicio do roteamento, é criado um nó correspondente em um grafo de rotas. Atribui-se então um peso para o nó, que corresponde ao número de vezes que aquele dado grupo se reuniu nas últimas T horas. O peso de cada nó V_i deve ser então normalizado pelo nó de maior peso: $W(V_i) = W(V_i)/max_i(W(V_i))$. A premissa é que grupos que se encontram com grande frequência tem maior probabilidade de se reunir novamente no futuro próximo e, dessa forma, o grupo com a maior frequência de encontros terá associado o valor 1. Entre cada par de nós, são criadas arestas de peso $W(E_{i,j}) = \#(V_i \cap V_j)/max(\#Vi, \#Vj)$, onde $\#(V_i \cap$



(a) (b)

Figura 4. **a)** Probabilidade de um grupo se reencontrar t horas após ter se encontrado pela primeira vez em $t = 0$. Linhas vermelhas representam períodos de 24 horas e linhas verdes períodos de 7 dias. **b)** Taxa de entrega média, para diferentes nós de origem, de Novembro de 2008 a Junho de 2009.

$V_j)$ é o número de pessoas comuns aos dois grupos e $max(\#Vi, \#Vj)$ é o número de componentes do maior dos dois grupos. Com isso, grupos com mais membros em comum passam a ter arestas de maior peso entre si. O peso das arestas está relacionado com a probabilidade de uma mensagem ser transmitida entre o par de grupos. Ao fim dessa inicialização obtém-se um grafo $G(V, E)$ em que V é o conjunto dos grupos encontrados e E é o conjunto de arestas que representam as interseções entre os grupos.

Com o objetivo de encontrar o caminho mais provável em $G(V, E)$, recalcula-se o peso de cada aresta $E_{i,j}$ como $W(E_{i,j}) = W(V_i) * W(V_j) * W(E_{i,j})$, onde V_i e V_j são os nós da aresta $E_{i,j}$. Com isso, considera-se a chance de ambos os grupos ligados pela aresta se reunirem e a chance da mensagem ser levada de um deles para o outro. O novo peso das arestas pode ser entendido como uma aproximação da probabilidade de uma mensagem ser propagada entre os membros do par de grupos no futuro próximo, considerando a probabilidade de reunião de cada um dos grupos e a possibilidade da mensagem ser levada de um grupo para o outro por algum usuário que é membro de ambos. Por fim, para computar as rotas mais prováveis de propagação da mensagem, assume-se independência entre as ocorrências das arestas e a probabilidade de uma rota R passa a ser calculada de acordo com a Eq. 1.

$$P(R) = \prod W(E_{i,j}), t.q. E_{i,j} \in R \tag{1}$$

Tirando proveito da propriedade de máxima verosimilhança de logaritmos pode-se obter o conjunto de arestas que maximizam a Eq. 1 de acordo com a Eq. 2.

$$arg.max(\prod W(E_{i,j})) = arg.max(log(\prod W(E_{i,j}))) = arg.max(\sum log(W(E_{i,j}))) \tag{2}$$

A Eq. 2 mostra que encontrar o caminho que maximiza o produto das probabilidades das arestas é equivalente a encontrar o maior caminho em um grafo onde o peso de cada aresta é igual ao oposto do logaritmo de sua probabilidade, problema conhecido como *Longest Path Problem*, que não apresenta solução em tempo polinomial. No

entanto, como as probabilidades das arestas assumem valores no intervalo $[0, 1]$, o logaritmo das arestas (Eq. 2) sempre resulta em números negativos, permitindo a aplicação de um algoritmo clássico de caminhos mínimos em grafos (como Djikstra), que pode ser resolvido em tempo polinomial. Dessa forma pode ser encontrado o caminho mais provável R no grafo de grupos.

Nesse ponto vale ressaltar que para encontrar a rota mais provável é necessário que a informação sobre os encontros dos grupos esteja centralizada em algum ponto. Isso não é problema em redes celulares D2D, já que esse processamento pode ser realizado na estação base. Para isso basta que os dispositivos atualizem de forma periódica (e.g. semanalmente) a estação base com os grupos aos quais pertenceram e o número de vezes que esses grupos se encontraram. Dessa forma, no momento em que um usuário desejar enviar um conteúdo para um determinado destinatário, a estação base envia para este usuário a rota grupo-a-grupo mais provável R. De posse da rota mais provável R, a cada encontro, um dispositivo que possui a mensagem a encaminha caso o dispositivo encontrado pertença a um dos grupos de R. Dessa forma a mensagem é propagada grupo-a-grupo, ao longo da rota R, até o destino.

Embora um caminho grupo-a-grupo seja o mais provável, não há garantia de que esse caminho de fato ocorra. Por isso foi adicionado ao algoritmo o parâmetro NRP. Com $NRP = 1$ apenas o caminho mais provável será utilizado para encaminhar as mensagens. Com $NRP = 2$, os dois caminhos, independentes em termos de arestas, serão utilizados para o encaminhamento das mensagens. O parâmetro NRP adiciona um *trade-off* entre taxa de entrega e sobrecarga da rede. Quanto menor NRP mais seletivo é o encaminhamento. A medida que o parâmetro aumenta, o algoritmo de roteamento tende a se assemelhar a uma inundação (*flooding*).

5. Experimentos

5.1. Metodologia

Para validar o algoritmo de encaminhamento proposto, realiza-se a comparação com o BubbleRAP [Hui et al. 2011], que se destacou como o mais eficiente em termos de sucesso na entrega das mensagens e baixa sobrecarga da rede. Nos experimentos realizados neste trabalho foram utilizados os parâmetros $T = 3$ semanas e $NRP = 1$. Avaliação do efeito da variação desses parâmetros é considerada nos trabalhos futuros (Seção 6).

5.1.1. Métricas

Com o objetivo de analisar comparativamente o desempenho do algoritmo proposto com relação ao BubbleRAP, que é o atual estado da arte, propõe-se a utilização as seguintes métricas:

- **Taxa de Entrega:** Avalia a porcentagem das mensagens que são entregues com sucesso ao longo do tempo t.
- **Número de Transmissões:** Mede a sobrecarga da rede, ou seja, o número de transmissões dispositivo-dispositivo que cada algoritmo realiza ao longo do tempo t.

Nos testes são sorteadas 500 tuplas $(origem, destino, tempo)$ e cada um dos algoritmos é executado com cada uma das tuplas sorteadas. Dessa forma, captura-se padrões

de comportamento diversificados ao longo dos traces, conferindo generalidade aos testes. Juntamente com os resultados dos algoritmos são plotados os resultados para uma inundação, em que um nó que possui a mensagem sempre a propaga para um nó que não a possui. A inundação consiste no limite superior para taxa de entrega e sobrecarga da rede. Cada campanha de 500 execuções é repetida oito vezes com diferentes sementes para geração das 500 tuplas aleatórias. Com isso, foram obtidos intervalos confiança de 95% para os testes.

5.1.2. BubbleRAP

O algoritmo BubbleRAP identifica comunidades sociais densamente interconectadas a partir do grafo de contatos agregados de todo o trace. Portanto, cada nó passa a pertencer a pelo menos uma comunidade. Atribui-se a nós que não estejam em nenhuma comunidade coletiva uma pseudo-comunidade de apenas um nó, para fins de encaminhamento. Além disso, a cada nó é atribuída uma medida de sua popularidade em relação a todos os nós do trace (*GlobalRank*) e em relação somente aos nós membros de sua comunidade (*LocalRank*).

Com isso o encaminhamento da origem até o destinatário final se dá de acordo com a seguinte política:

- A cada encontro, um dado nó transmite a mensagem se o nó encontrado tiver um *GlobalRank* maior que o seu próprio ou se o nó encontrado pertencer a uma comunidade em conjunto com o destinatário final.
- Uma vez que a mensagem chega a algum nó membro da comunidade à qual o destinatário final pertence, o encaminhamento fora desta comunidade deixa de ocorrer.
- Uma vez que a mensagem chegou na comunidade destino, o encaminhamento passa a ocorrer caso o *LocalRank* do nó encontrado seja maior que o do nó que possui a mensagem ou caso o nó encontrado seja o destinatário final da mensagem.

Para fins de comparação, o BubbleRAP foi implementado utilizando os parâmetros reportados em [Hui et al. 2011] e que geraram os melhores resultados em termos de taxa de entrega e sobrecarga da rede. O cálculo do *GlobalRank* e do *LocalRank* foi realizado utilizando a técnica C-Window que, segundo [Hui et al. 2011], melhor aproximou métrica de centralidade de um nó em seus experimentos.

5.2. Resultados

As Figuras 5 e 6 apresentam, em escala logarítmica, os resultados em termos de taxa de entrega e sobrecarga da rede com transmissões em função do tempo após o início da propagação da mensagem.

Os resultados para o *trace* MIT Reality Mining, na Figura 5, mostram que nas primeiras horas após o início da propagação o GROUPS-NET apresenta uma taxa de entrega sutilmente maior e nas horas finais Bubble Rap ultrapassa o GROUPS-NET. Além disso, ao longo de todo o tempo de propagação, o GROUPS-NET apresentou uma sobrecarga pouco maior que a do Bubble Rap. Vale ressaltar que o *trace* MIT Reality Mining é muito particular, pois todos os nós monitorados residem no mesmo local. Logo, espera-se que os laços sociais desse subconjunto dos usuários sejam mais fortes do que a média, fato esse que beneficia o Bubble Rap, que usa a estrutura social estática e a popularidade dos nós

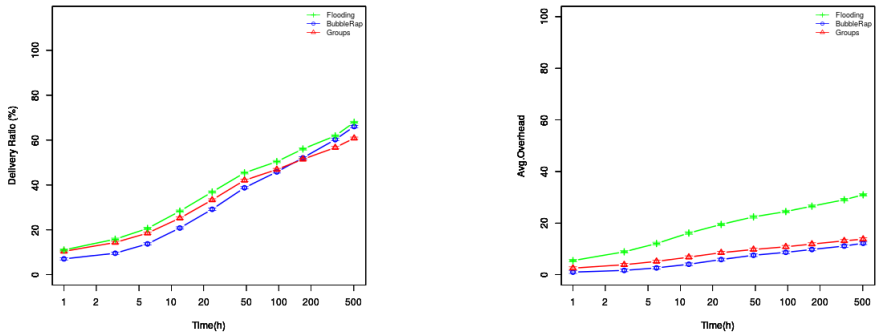


Figura 5. MIT Reality (80 nós): Taxa de entrega e número de transmissões.

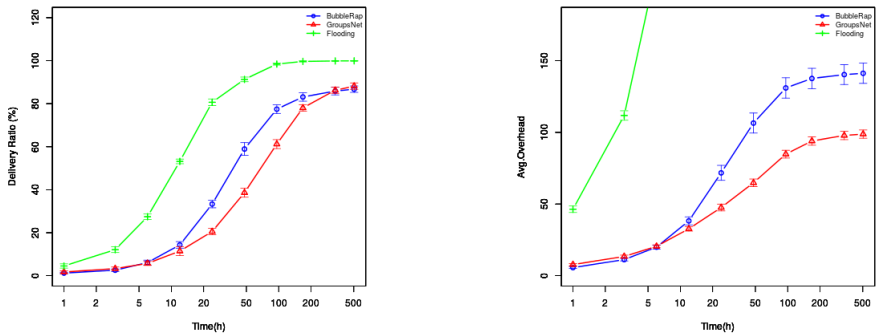
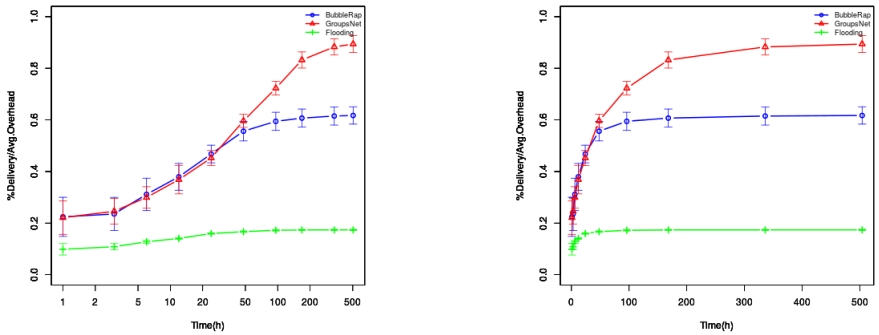


Figura 6. Dartmouth (1200 nós): Taxa de entrega e número de transmissões.

para definir sua política de encaminhamento. Esse fato que motivou a realização de testes em um trace de larga escala como o Dartmouth. Não obstante, é importante enfatizar que, mesmo nesse cenário específico, o GROUPS-NET apresentou um desempenho bastante similar ao do Bubble Rap, com a vantagem de não requerer detecção de comunidades.

A Figura 6 mostra os resultados obtidos para o trace Dartmouth. No *trace* Dartmouth, 1200 alunos de um campus universitário foram monitorados, sendo portanto um trace de mais larga escala e menos específico que o MIT Reality Mining. Nesse caso o GROUPS-NET obteve um desempenho consideravelmente melhor que o Bubble RAP. No periodo entre 24 e 96 horas após o inicio da transmissão o Bubble Rap obtém uma maior taxa de entrega, sendo ultrapassado pelo GROUPS-NET nas horas seguintes e até o fim do período de três semanas. Em relação à sobrecarga da rede, após a sexta hora o Bubble Rap passa a transmitir muito mais mensagens que o GROUPS-NET, apresentando uma sobrecarga em média 50% maior que a do GROUPS-NET nas horas seguintes. Para mil mensagens $(origem, destino)$ no trace Dartmouth, isso representa uma economia de cerca de 60 mil transmissões dispositivo-dispositivo.

Definindo-se uma métrica de benefício-custo como a razão entre a taxa de entrega e a sobrecarga da rede a cada instante de tempo, o GROUPS-NET mantém um desempenho equivalente ou melhor que o Bubble RAP em todos os momentos após o início das transmissões. Como mostrado na figura 7, após as primeiras horas de transmissão, o desempenho do GROUPS-NET, em termos de benefício-custo, chega a ser 80% maior. Como mencionado anteriormente, o GROUPS-NET ainda tem a vantagem de utilizar detecção de grupos e não detecção de comunidades, o que é mais viável na prática.



(a) Escala log (b) Escala regular

Figura 7. Comparação da relação benefício-custo dos protocolos Bubble Rap e Groups ao longo do tempo no trace Dartmouth.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi descrito o GROUPS-NET, um algoritmo de roteamento oportunístico ciente de encontros de grupos para redes celulares D2D. O algoritmo possui a vantagem de não necessitar de detecção distribuída de comunidades, além de considerar a dinamicidade de encontros sociais em contraste com estratégias anteriores, que consideram a estrutura social estática de comunidades.

O GROUPS-NET foi comparado com o algoritmo do estado da arte Bubble Rap em um cenário particular e de pequena escala, por meio do trace MIT Reality, apresentando desempenho similar. Já em um cenário de larga escala e menos específico, como o *trace* de Dartmouth, o GROUPS-NET demonstrou um desempenho superior, realizando 40% menos transmissões para atingir a mesma taxa de entrega. Foi analisada a relação benefício-custo dos dois algoritmos e para todos os tempos de transmissão o GROUPS-NET apresentou melhor benefício-custo.

Como trabalho futuro destaca-se a utilização de informações de geo-localização e pontos de interesse combinados com a ciência de encontros de grupos para aprimorar os resultados do algoritmo. Além disso, pode ser feito um estudo de como a variação no período de tempo a ser considerado antes da transmissão e o número de caminhos redundantes (parâmetros T e NRP) afetam o desempenho do algoritmo em cada cenário de testes. Finalmente, destaca-se a possibilidade de adaptar o GROUPS-NET para atuar em cenários que a mesma mensagem tem múltiplos destinatários.

Referências

Bauza, R., Gozalvez, J., and Sanchez-Soriano, J. (2010). Road traffic congestion detection through cooperative vehicle-to-vehicle communications. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 606–612. IEEE.

Calabrese, F., Pereira, F. C., Di Lorenzo, G., Liu, L., and Ratti, C. (2010). The geography of taste: analyzing cell-phone mobility and social events. In *Pervasive computing*, pages 22–37. Springer.

Eagle, N. and Pentland, A. (2006). Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268.

Fodor, G., Dahlman, E., Mildh, G., Parkvall, S., Reider, N., Miklós, G., and Turányi, Z. (2012). Design aspects of network assisted device-to-device communications. *Communications Magazine, IEEE*, 50(3):170–177.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3):75–174.

Gonzalez, M. C., Hidalgo, C. A., and Barabasi, A.-L. (2008). Understanding individual human mobility patterns. *Nature*, 453(7196):779–782.

Gregory, S. (2010). Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018.

Henderson, T., Kotz, D., and Abyzov, I. (2008). The changing usage of a mature campus-wide wireless network. *Computer Networks*, 52(14):2690–2712.

Hui, P., Crowcroft, J., and Yoneki, E. (2011). Bubble rap: Social-based forwarding in delay-tolerant networks. *Mobile Computing, IEEE Transactions on*, 10(11):1576–1589.

Hui, P., Yoneki, E., Chan, S. Y., and Crowcroft, J. (2007). Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, page 7. ACM.

Lei, L., Zhong, Z., Lin, C., and Shen, X. (2012). Operator controlled device-to-device communications in lte-advanced networks. *IEEE Wireless Communications*, 19(3):96.

Li, Y., Wu, T., Hui, P., Jin, D., and Chen, S. (2014). Social-aware d2d communications: qualitative insights and quantitative analysis. *Communications Magazine, IEEE*, 52(6):150–158.

Lindgren, A., Doria, A., and Schelén, O. (2003). Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20.

Lu, J. and Cao, L. (2003). Congestion evaluation from traffic flow information based on fuzzy logic. In *Intelligent Transportation Systems, 2003.*, volume 1, pages 50–53. IEEE.

Nguyen, N. P., Dinh, T. N., Tokala, S., and Thai, M. T. (2011). Overlapping communities in dynamic networks: their detection and mobile applications. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 85–96. ACM.

Nunes, I., Vaz de Melo, P., and A.F. Loureiro, A. (2016). Group mobility: Detection, tracking and characterization. In *IEEE ICC 2016 International Conference on Communications (ICC'16 SAC-8 SN)*, Kuala Lumpur, Malaysia.

Palla, G., Barabási, A.-L., and Vicsek, T. (2007). Quantifying social group evolution. *Nature*, 446(7136):664–667.

Palla, G., Derényi, I., Farkas, I., and Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818.

Peel, L. (2010). Estimating network parameters for selecting community detection algorithms. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1–8. IEEE.

Silva, T. H., Vaz de Melo, P. O., Almeida, J. M., Salles, J., and Loureiro, A. A. (2014). Revealing the city that we cannot see. *ACM Transactions on Internet Technology (TOIT)*, 14(4):26.

Xu, H., Hu, Y., Wang, Z., Ma, J., and Xiao, W. (2013). Core-based dynamic community detection in mobile social networks. *Entropy*, 15(12):5419–5438.

Analisando a Capacidade de Descarregamento de Redes Móveis por meio de Redes Oportunísticas *

Vinícius F. S. Mota[1], Henrique Moura[1], Vinicius F. Silva[1], Daniel F. Macedo[1], Yacine Ghamri-Doudane[2], José Marcos Silva Nogueira[1]

[1] Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte – Brazil

[2]L3i Lab – Université de La Rochelle
La Rochelle – France

{vfsmota, henrique, viniciusfs, damacedo, jmarcos}@dcc.ufmg.br

yacine.ghamri@univ-lr.fr

Abstract. *Device-to-Device (D2D) communication paradigm arises as an approach to relieve data traffic from mobile Internet Service Providers. In D2D, end-user devices can help operators to improve cell coverage, spectrum efficiency and to offload data. This work discuss the feasibility of D2D offloading through a quantitative evaluation of three applications: (i) Opportunistic Relaying (OpR), nodes in the opportunistic mode attempt to forward their data through relay nodes instead of sending them to the infrastructure; (ii) Cache-and-Forward (CaF), relay nodes forward all data received to all opportunistic nodes within a threshold delay; and (iii) Relay-as-Cache (RaC), opportunistic nodes seek for a determined content in the relay nodes cache. Our simulations show that in scenarios with high mobility and high delay tolerance relay nodes can forward up to 70% of the traffic in OpR, 94% in CaF and up to 35% in RaC.*

Resumo. *O paradigma de comunicação dispositivo-a-dispositivo surge como uma abordagem para aliviar o tráfego de dados de prestadores de serviços de Internet móvel. Nas redes D2D, dispositivos de usuário podem ajudar as operadoras a melhorar a cobertura celular, a eficiência do espectro e o descarregamento de dados. Este trabalho discute a viabilidade do descarregamento de dados móveis por meio de redes D2D através de um estudo quantitativo de três aplicações D2D: (i) Opportunistic Relaying (OpR), dispositivos em modo oportunista transmitem suas mensagens de dados para nós retransmissores em vez de enviá-las diretamente para a infraestrutura; (ii) Cache-and-Forward (CaF), nós retransmissores encaminham todos os conteúdos recebidos para todos os nós oportunistas encontrados dentro de um prazo limite; (iii) Relay-as-Cache (RaC), nós oportunistas procuram por um determinado conteúdo no cache dos nós retransmissores. Simulações mostram que em cenários com alta mobilidade e com alta tolerância a atrasos os nós retransmissores podem transmitir até 70% do tráfego de dados na aplicação OpR, 94% na CaF e até 35% na RaC.*

*Os autores gostariam de agradecer à CAPES, CNPq e FAPEMIG pelo apoio financeiro parcial a este trabalho.

1. Introdução

A evolução de diferentes tecnologias utilizadas na Internet possibilitou a criação de novas aplicações, tais como mensagens instantâneas, VoIP e *streaming* de vídeo. Por essa razão, a demanda por maiores larguras de banda tem crescido de forma exponencial a cada ano [Cisco 2015]. Os provedores de Internet têm enfrentado gargalos de tráfego cada vez mais constantes para atender com qualidade aos requisitos de seus usuários. Uma solução para amenizar o problema é mover o tráfego de dados das redes celulares para outras interfaces de comunicação. Esta estratégia é conhecida como Descarregamento de Dados das Redes Móveis (*mobile data offloading*).

Soluções existentes para descarregamento de dados requerem equipamentos especiais ou uma reformulação da rede, o que pode implicar em mudanças significativas em sua infraestrutura. De fato, a indústria e a academia têm procurado realizar o descarregamento de dados provenientes das redes celulares por meio do uso de *femtocélulas* [Haldar et al. 2013], redes Wi-Fi públicas [Mota et al. 2013] e, mais recentemente, redes oportunísticas dispositivo-a-dispositivo (D2D) [Doppler et al. 2009].

No descarregamento oportunístico de dados, um subconjunto dos nós da rede atua como retransmissores para um outro subconjunto, que acessará a rede de forma oportunística por meio destes retransmissores. Por exemplo, em grandes aglomerações (*shows*, eventos esportivos, etc.), dispositivos com conectividade comprometida podem enviar ou obter conteúdo oportunisticamente por meio dos dispositivos retransmissores.

Neste trabalho focamos na análise de cenários em que redes D2D oportunísticas podem realizar o descarregamento de dados provenientes de redes celulares. Para atingir nosso objetivo, implementamos três aplicações, nas quais o descarregamento oportunístico de dados melhora a utilização de recursos da rede:

(1) *Opportunistic Relaying* - OpR: Nós retransmissores encaminham todas as requisições provenientes de nós oportunistas. Deste modo, nós oportunistas deixam a rede celular, reduzindo assim o tráfego de sinalização [Choi et al. 2014];
(2) *Cache-and-Forward* - CaF: Nós retransmissores encaminham todo conteúdo recebido para todos os nós oportunistas encontrados dentro de um intervalo de tempo.
(3) *Relay-as-Cache* - RaC: Um nó oportunista busca por um determinado conteúdo em nós retransmissores com uma tolerância de atraso pré-definida.

Para a seleção dos subconjuntos de nós retransmissores e nós oportunistas, cada aplicação foi analisada utilizando duas estratégias: *Random*, onde os nós são escolhidos aleatoriamente como retransmissores, e *OppLite*, uma ferramenta que utiliza características da rede para decidir quais nós se tornarão retransmissores [Mota et al. 2014].

A principal contribuição deste trabalho é a extensa avaliação quantitativa dos cenários citados acima. Tal avaliação mostra que a tolerância ao atraso, bem como a disposição dos usuários serem retransmissores, são pontos chave para obter altas taxas de descarregamento de dados.

O restante deste trabalho está organizado da seguinte forma: A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a arquitetura do sistema, detalhando os algoritmos para seleção de nós retransmissores e cada aplicação desenvolvida. A metodologia das análises é apresentada na Seção 4 e os resultados são discutidos na Seção 5. Por fim, a Seção 6 conclui este trabalho.

2. Trabalhos Relacionados

No descarregamento oportunista, um subconjunto de dispositivos (ou nós)[1] desempenham o papel de retransmissores entre outros dispositivos e a infra-estrutura. Visando o estabelecimento de comunicação "*ad hoc*" entre dispositivos próximos da rede celular de modo transparente, o consórcio 3GPP propôs o padrão *Device-to-Device Proximity Service* (*D2D ProSe*) [3GPPP 2013]. *D2D ProSe* reutiliza a interface de rádio LTE para a comunicação entre dispositivos, de modo que a infra-estrutura possa participar na configuração da sessão, na conexão e na criação de rotas entre dispositivos.

A viabilidade da utilização de dispositivos terminais como *cache* foi estudada em [Valerio et al. 2015]. *Valerio et al.* analisaram a probabilidade de encontrar vídeos de usuários do YouTube no *cache* do dispositivo dos usuários. Os autores mostraram que a requisição de vídeo segue uma distribuição de probabilidade Zipf e o *cache* local pode responder até 21% das requisições de outros usuários.

A maioria dos trabalhos relacionados a técnicas de descarregamento oportunista focam na proposta de algoritmos para selecionar nós retransmissores [Rebecchi et al. 2015].

Em [Doppler et al. 2009], os autores focaram nas mudanças necessárias na infra-estrutura celular para implantar comunicação D2D e mostraram que o uso de comunicação D2D pode aumentar a largura de banda dos participantes da rede. No entanto, nesta proposta os nós retransmissores são escolhidos aleatoriamente. Já no *VIP*, proposto em [Barbera et al. 2011], os nós retransmissores são escolhidos baseando-se em métricas sociais, como centralidade, *betweenness* e *Page Rank*. No *VIP*, os nós retransmissores replicam oportunisticamente dados recebidos da infra-estrutura para outros nós. De forma semelhante, um arcabouço com várias estratégias para decidir *quando* e *para quem* um conteúdo deve ser enviado, batizado de *Push-and-Track*, foi proposto em [Whitbeck et al. 2012]. No *Push-and-Track*, o nó retransmissor replica o conteúdo para outros nós da rede. Por fim, o *OppLite* é um arcabouço que considera a disposição do usuário para se tornar um retransmissor ou um nó oportunista com base em um conjunto de propriedades das redes e dos dispositivos [Mota et al. 2014].

Este trabalho difere do estado da arte pois implementa e analisa quantitativamente os diversos cenários de descarregamento de dados móveis propostos na literatura. Desta forma, podemos discutir em quais condições o descarregamento oportunístico é benéfico tanto para o usuário quanto para os provedores de Internet móvel.

3. Arquitetura do Sistema

Para avaliar a viabilidade de descarregamento D2D sob vários cenários de aplicação, desenvolvemos um sistema que nos permite implementar separadamente aplicações e algoritmos para selecionar dispositivos como retransmissor. A Figura 1 retrata a arquitetura desse sistema. Cada aplicação, a qual define as regras de descarregamento, é executada no topo de um algoritmo utilizado para selecionar o nó retransmissor adequado. Por sua vez, cada algoritmo implementa as regras que definem como um nó se torna um retransmissor

[1]A nomenclatura *dispositivos* ou *nós* é usada de modo intercambiável ao longo do texto. O primeiro refere-se ao uso ou posse do dispositivo, enquanto *nós* se refere a vértices do grafo formado pela rede.

e qual interface de rede pode ser utilizada pela aplicação. As subseções a seguir detalham cada algoritmo e as aplicações utilizadas na nossa análise.

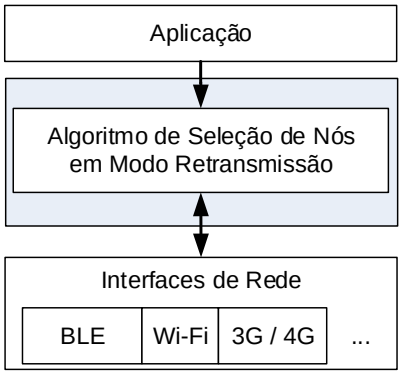


Figura 1. Arquitetura genérica de um sistema de descarregamento de dados.

3.1. Algoritmos de Seleção de Retransmissores

Utilizamos duas estratégias para selecionar nós como retransmissores: *Random* e *OppLite* [Mota et al. 2014]. Na primeira estratégia, os nós retransmissores e os nós oportunistas são escolhidos aleatoriamente, enquanto o *OppLite* realiza a tomada de decisão utilizando múltiplos critérios. Em comum, ambos os algoritmos permitem que o usuário determine sua disposição em se tornar um retransmissor e a tolerância de atraso (τ) suportada pelos nós oportunistas por meio de parâmetros configuráveis.

Sejam $\Gamma_{relay}, \Gamma_{opp} \in [0, 1]$ limiares configuráveis que representam a disposição do usuário para cooperar e se tornar um nó retransmissor ou um nó oportunista, respectivamente. Valores próximos a zero indicam alta cooperação, enquanto valores próximos a um indicam baixa vontade de cooperação com a rede.

3.1.1. *Random* - Seleção Aleatória

No *Random*, os nós retransmissores e oportunistas são escolhidos aleatoriamente. Sejam variáveis aleatórias $X, Y \in [0, 1]$ que seguem uma distribuição uniforme. Um dispositivo torna-se um retransmissor se $X > \Gamma_{relay}$ e oportunista se $Y > \Gamma_{opp}$. A fim de permitir que os nós mudem suas regras e seu modo de operação, o *Random* elege novos valores para X e Y a cada 60 segundos.

3.1.2. *OppLite* - Seleção baseada em multi-critérios

O *OppLite* é um arcabouço de tomada de decisão multi-critério com base na teoria da utilidade, que permite alternar entre os modos de infra-estrutura e oportunista com base apenas nas decisões locais realizadas nos dispositivos [Mota et al. 2014]. O *OppLite* usa o número de vizinhos, a vida útil da bateria e a intensidade do sinal como critérios para tomar a decisão de enviar ou requisitar dados diretamente para a infraestrutura ou por meio de comunicação oportunista.

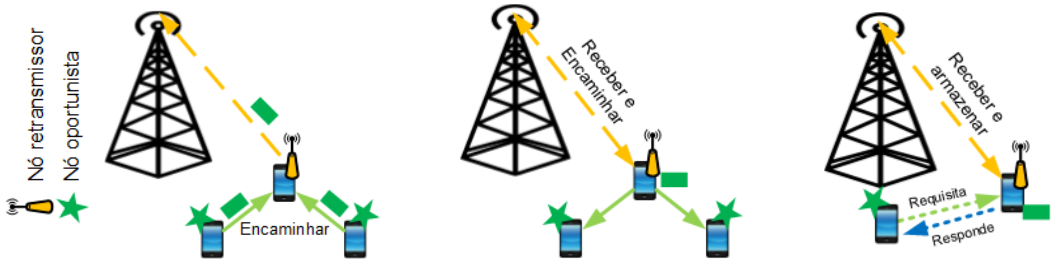
Cada critério (x) é quantificado por meio da Função sigmoide (1), sendo o centro (x_t) e o grau de inclinação (α) parâmetros do algoritmo. A utilidade multi-critério é obtida pela Equação (2), sendo $w_i \in [0, 1]$ o peso para cada critério e $\sum_i w_i = 1$. No *OppLite*, um dispositivo torna-se um retransmissor se a utilidade agregada $U > \Gamma_{relay}$.

$$u(x) = \frac{1}{1 + e^{\alpha(x_t - x)}} \tag{1}$$

$$U(x_1 \dots x_n) = \prod_{i=1}^{n} [u(x_i)]^{w_i} \tag{2}$$

3.2. Aplicações de Descarregamento

Desenvolvemos três aplicações para medir a capacidade de descarregamento de dados da rede. Estas aplicações são ilustradas na Figura 2 e descritas nas subseções a seguir.



(a) OpR **(b)** CaF **(c)** RaC

Figura 2. Cenários de aplicação de descarregamento de dados móveis.

3.2.1. Opportunistic Relaying (OpR)

Nesta aplicação, mostrada na Figura 2a, os nós oportunistas deixam a infraestrutura da rede para tentar o encaminhar suas mensagens através de nós retransmissores. Após a criação de uma mensagem, um nó encaminha a mensagem com base no seu modo de comunicação. Um nó oportunista tenta enviar a mensagem através de um nó retransmissor. Caso o nó oportunista não encontre um nó retransmissor adequado até um atraso tolerável (τ), ele envia suas mensagens diretamente para a infraestrutura, ou seja, volta para o modo padrão.

Este caso pode representar situações em que nós oportunistas estejam em redes de largura de banda inferior ou congestionadas. Embora nesta aplicação o tráfego ainda seja encaminhado através da rede celular, esta aplicação pode ajudar a aliviar o congestionamento de sinalização. Por exemplo, nas redes participativas de sensores, os dispositivos reúnem dados (usando seus sensores embutidos) e transmitem esses dados para a "nuvem" mantendo constantemente o envio de pequenos pacotes para a rede. Embora a quantidade de dados seja pequena, vários dispositivos enviando pacotes de forma contínua podem causar congestionamento na sinalização [Choi et al. 2014].

3.2.2. Cache-and-Forward (CaF)

A arquitetura *Cache and Forward* para roteadores de grande capacidade foi proposta por [Paul et al. 2008]. Consideramos que os dispositivos dos usuários podem atuar como

caches potenciais na rede. Esta aplicação modela aplicações de requisições HTTP, em que os usuários solicitam o conteúdo para a "nuvem". Esta aplicação pode representar uma atualização de software ou informações enviadas de algum canal, evitando a transmissão de dados redundantes a partir dos provedores de Internet.

A Figura 2b ilustra o funcionamento do CaF. Os nós no modo padrão ou no modo retransmissão requisitam conteúdos usando um identificador para cada conteúdo e a "nuvem" responde com o conteúdo solicitado. Os nós retransmissores armazenam o conteúdo em seus respectivos *buffers* e marcam quando o conteúdo foi recebido. No CaF, os nós retransmissores encaminham cada conteúdo em seus *buffers* para todos os nós oportunistas encontrados até o tempo de vida do conteúdo expirar, sendo o tempo de vida definido pelo parâmetro τ.

3.2.3. Relay-as-Cache (RaC)

O RaC, mostrado na Figura 2c, também utiliza os dispositivos retransmissores como *cache* da rede. No RaC, nós oportunistas solicitam um conteúdo aos vizinhos retransmissores. Esta aplicação pode ajudar a aliviar o tráfego de dados em situações de multidão onde várias pessoas estão à procura de conteúdo idêntico, tais como torcedores em um jogo de futebol buscando a repetição de um gol [Valerio et al. 2015].

No RaC, todos os nós são elegíveis para solicitar conteúdo. Um nó envia mensagens de solicitação contendo a identificação do conteúdo. Nós em modo *padrão* ou *retransmissão* enviam as requisições através da rede celular. Por sua vez, a "nuvem" responde com o conteúdo requisitado. Nós retransmissores armazenam todo o conteúdo recebido em seus *buffers*, funcionando como *cache* de armazenamento, aceitando requisições de nós em modo oportunista.

Um nó oportunista busca um determinado conteúdo no *cache* de nós retransmissores. Esse processo se repete até que algum retransmissor responda ou até que expire a tolerância de atraso máximo definida por τ. Após a tolerância de atraso expirar, RaC utiliza a rede infra-estruturada para fazer a requisição. Um *acerto* ocorre quando um nó oportunista encontra o conteúdo solicitado no *cache* do nó retransmissor.

4. Metodologia

Implementamos as aplicações e os algoritmos de seleção de nós retransmissores no *Opportunistic Networking Environment* (ONE) [Keränen et al. 2009]. Estendemos o ONE para suportar redes infra-estruturadas e permitir que o algoritmo *OppLite* tenha acesso às propriedades da rede: sinal da rede celular, a vizinhança do nó e a bateria remanescente.

A aplicação OpR visa reduzir o número de nós enviando mensagens diretamente para a infra-estrutura. Já as aplicações CaF e RaC visam reduzir a utilização da infra-estrutura em solicitações de conteúdo previamente requeridos por um ou mais usuários. Portanto, as aplicações quantificam o número de mensagens entregues de modo oportunista usando a métrica *taxa de descarregamento*:

$$\text{Taxa de descarregamento} = \frac{\text{\# Msgs Enviadas de forma oportunista}}{\text{\# Msgs Enviadas de forma oportunista} + \text{\# Msgs Enviadas pela infra-estrutura}}$$

Exclusivamente no caso da aplicação CaF, analisamos o número médio de mensagens enviadas por cada nó retransmissor.

4.1. Configuração

Os nós possuem duas interfaces de comunicação: celular 3G e IEEE 802.11. Os nós podem comunicar diretamente entre si usando a interface IEEE 802.11 em modo "*ad hoc*". A interface Wi-Fi tem uma taxa de 2Mbps para a comunicação "*ad hoc*", que é uma taxa compatível com as taxas comuns em redes IEEE 802.11. A interface de rede celular tem uma taxa de 100kBps (de acordo com medições em [Whitbeck et al. 2012]). Cada nó possui 500 MB de espaço em armazenamento. Cada mensagem tem entre 100KB e 1MB de carga útil, representando texto e/ou imagens. Cada teste foi executado 15 vezes e os resultados possuem 95% de intervalo de confiança.

O tempo de geração inter-mensagens ou inter-requisições de cada nó na rede segue uma distribuição Weibull com declividade $k = 21,99$ e escala $\lambda = 1,429$. Além disso, a popularidade do conteúdo é uma métrica importante para mecanismos de *cache*. A solicitação de conteúdos na aplicação *Relay-as-Cache* segue uma distribuição Zipf com expoente $\rho = 1,47$, similar com os resultados apresentados em [Valerio et al. 2015]. A fim de refinar nossas avaliações, utilizamos dois *traces* reais de contatos de rede:

- **INFOCOM**: *trace* de contato de três dias com 20 nós estáticos e 78 usuários voluntários usando iMotes durante a edição de 2006 da confêrencia INFOCOM [Haggle 2013]. Utilizamos os valores do primeiro dia da conferência, entre 12:00 e 18:00. As aplicações param de gerar mensagens às 17:00, assegurando tempo suficiente para a entrega das mensagens.
- **ROLLERNET**: 62 voluntários carregam iMotes durante a turnê *Pari-Roller*, que reúne milhares de pessoas para patinar pelas ruas de Paris, em 2006 [Tournoux et al. 2009]. Uma vez que este *trace* registra todos os dispositivos Bluetooth ao alcance, ele contém 1112 dispositivos no total. A duração do rastreamento de contato é de cerca de três horas. As aplicações geram mensagens ou requisitam conteúdos entre 30 e 160 minutos.

O conjunto de critérios em *OppLite* foi configurado conforme descrito em [Mota et al. 2014]. Consideramos que o usuário tem alta disponibilidade de se tornar um nó oportunista ($\Gamma_{opp} = 0.01$), pois há vantagens para estes nós, como economia de energia por exemplo. Os parâmetros comuns entre o *OppLite* e o *Random* - a disposição do usuário em se tornar um nó retransmissor (Γ_{relay}) e a tolerância de atraso (τ) - foram combinados usando os seguintes valores:

$$\Gamma_{relay} = [0,\ 0.2\ ,\ 0.5\ ,\ 0.7] \quad \tau = [1, 100, 600, 1200]\text{s}$$

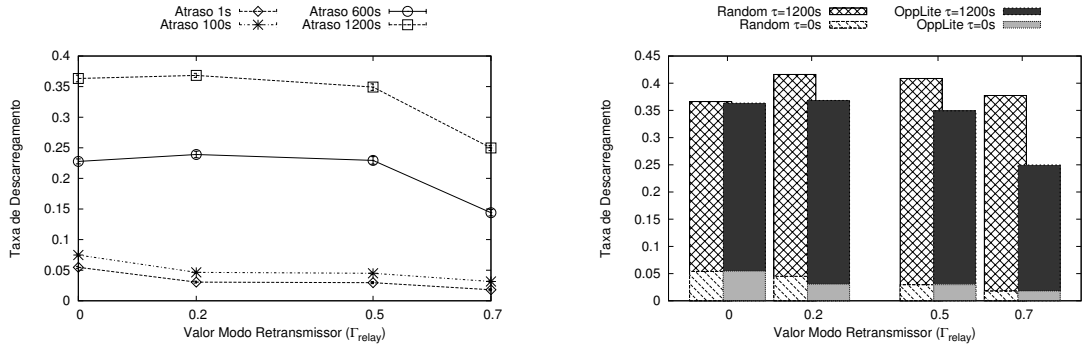
5. Análises das Aplicações

Esta seção avalia as aplicações OpR, CaF e RaC com os algoritmos de seleção de nós retransmissores $Random$ e $OppLite$ descritos na Seção 3. O valor do modo retransmissor (Γ_{relay}) indica a disposição dos nós se tornarem retransmissores. Quando $\Gamma_{relay} = 0$, todos os nós estão dispostos a se tornarem retransmissores e quando $\Gamma_{relay} = 1$ nenhum nó se tornará retransmissor. Nas análises de todas as aplicações, mostramos a tolerância ao atraso utilizando o algoritmo *OppLite* e mostramos uma comparação do mesmo com o *Random*. Deste modo, podemos discutir o impacto da utilização de um algoritmo inteligente em relação à escolha aleatória de nós como retransmissores.

5.1. Análise do *Opportunistic Relaying - OpR*

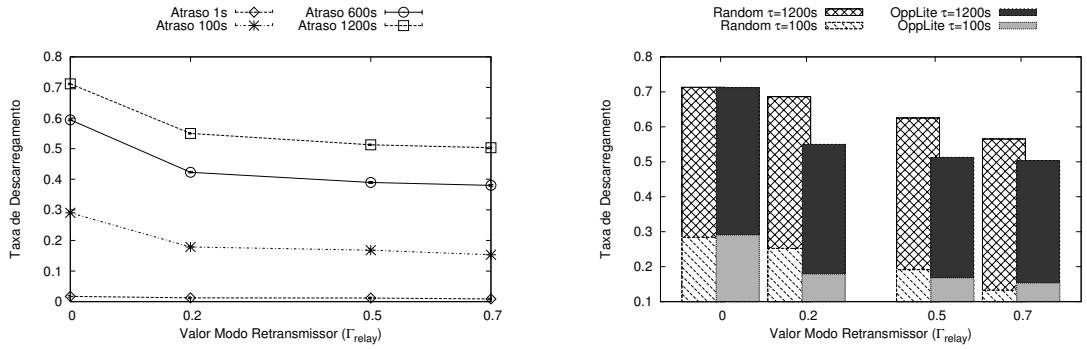
No OpR, os nós oportunistas devem transmitir suas mensagens através de nós retransmissores para reduzir o congestionamento de sinalização na rede infra-estruturada. Se o nó oportunista não encontra um nó retransmissor até uma tolerância de atraso τ, a aplicação encaminha a mensagem para infraestrutura. O descarregamento ocorre toda vez que um nó oportunista encaminha uma mensagem por algum nó retransmissor.

As Figuras 3 e 4 mostram a taxa do descarregamento de dados para valores crescentes de Γ_{relay} com diferentes valores de tolerância de atraso (τ).



(a) Tolerância de atraso (*OppLite*) **(b)** *Random* vs *OppLite*

Figura 3. OpR: Taxa de descarregamento de dados no cenário INFOCOM



(a) Tolerância de atraso (*OppLite*) **(b)** *Random* vs *OppLite*

Figura 4. OpR: Taxa de descarregamento de dados no cenário ROLLERNET

No cenário INFOCOM, foram criadas 5700 mensagens. A Figura 3a mostra que com a tolerância de atraso igual a zero, o OpR, utilizando o *OppLite*, descarrega 6% das mensagens quando todos os nós viram retransmissores ($\Gamma_{relay} = 0$) e 3% quando poucos nós se tornam retransmissores ($\Gamma_{relay} = 0.7$).

A Figura 3b mostra a relação da taxa de descarregamento para uma tolerância de atraso τ de zero e de 1200 segundos (para melhor legibilidade omitimos os outros valores de atraso). Na aplicação *OpR*, *OppLite* e *Random* têm taxa de descarregamento semelhante ($\tau = 0s$), pois os nós oportunistas precisam ter vizinhos retransmissores. Para $\tau = 1200s$, escolher nós retransmissores aleatoriamente tem um melhor desempenho.

No cenário ROLLERNET, apenas os 62 dispositivos pertencentes ao experimento ROLLERNET podem criar mensagens, com isto, foram criadas 2390 mensagens. A Fi-

gura 4a mostra a influência da tolerância de atraso no descarregamento das mensagens. Observamos um comportamento constante para uma tolerância de atraso igual a zero, explicada pelo fato dos nós terem tempo de contato mais curtos em tal cenário. No entanto, até 50% e 65% das mensagens podem ser retransmitidas, quando o OpR tolera 600 ou 1200 segundos de atraso, respectivamente, para $\Gamma_{relay} \geq 0.5$.

A Figura 4b mostra o desempenho do OpR com os algoritmos *Random* e *OppLite* e tolerância de atraso de 100 e 1200 segundos no cenário ROLLERNET. Neste cenário, quando todos os nós são retransmissores, o *OpR* pode descarregar 74% do tráfego quando a aplicação tolera até 20 minutos de atraso. Ao diminuir a disposição dos nós para se tornarem retransmissores, $\Gamma_{relay} \geq 0.2$, o *Random* tem melhor desempenho pois seleciona um subconjunto maior de nós como retransmissores.

Em ambos os cenários, o descarregamento máximo é obtido quando os nós oportunistas suportam atrasos maiores, pois estes aguardam por um tempo maior um contato com um nó retransmissor. Além disto, diminuir a disposição do usuário em cooperar com a rede de $\Gamma_{relay} = 0$ para $\Gamma_{relay} = 0.7$ causa uma redução média de 30% na taxa de descarregamento de dados.

5.2. Análise do *Cache and Forward - CaF*

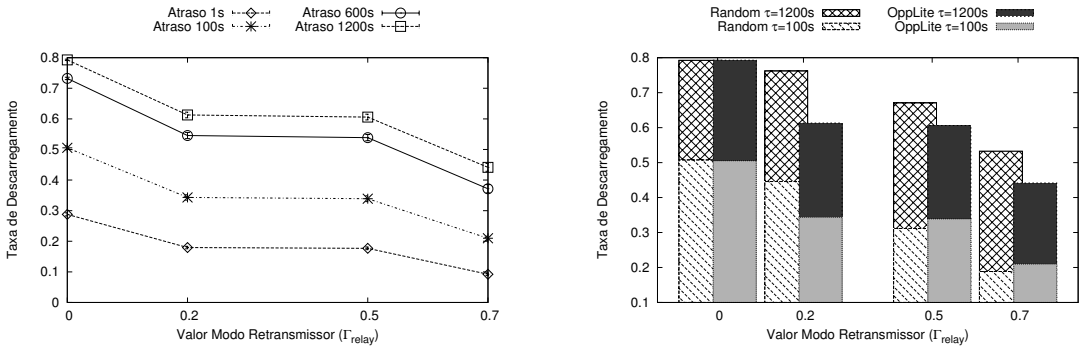
No CaF, o descarregamento dos dados ocorre sempre que um retransmissor tem a oportunidade de replicar uma mensagem para nós em modo oportunístico. O CaF busca maximizar o encaminhamento de mensagens pelos nós retransmissores.

No cenário do INFOCOM, os nós requisitaram 865 mensagens diretamente à infraestrutura. A Figura 5 mostra a taxa de descarregamento do CaF com diferentes valores de tolerância de atraso (τ). No CaF, τ representa o TTL (*time-to-live*) de cada mensagem recebida e, com isso, quanto tempo um nó retransmissor encaminha uma determinada mensagem para nós oportunistas. Portanto, valores elevados de τ acarretam em maiores taxas de descarregamento. De fato, como mostrado na Figura 5a, quando todos os nós estão em modo retransmissor ($\Gamma_{relay} = 0$), replicações de mensagem representam de 28% até 79% do tráfego de dados da rede com atrasos de tolerância de 1 e 1200 segundos, respectivamente. A taxa de descarregamento cai para 9% e 44% para $\tau = 1s$ e $\tau = 1200s$, respectivamente, quando $\Gamma_{relay} = 0.7$.

Com o incremento de τ de 1 a 1200 segundos, houve um aumento de 182% na taxa de descarregamento para todos os valores de Γ_{relay} em média. Essa melhoria é de apenas 12% em média ao incrementar τ de 600 para 1200 segundos.

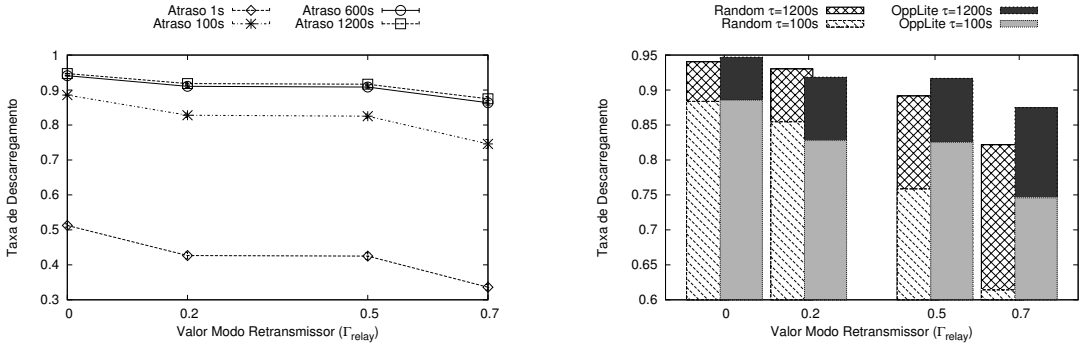
A Figura 5b mostra um comparativo do CaF entre os algoritmos *Random* e *OppLite* com 100 e 1200 segundos de tolerância de atraso. O CaF com *OppLite* supera a taxa de descarregamento do *Random* quando $\Gamma_{relay} \geq 0.5$ com valores reduzidos de tolerância de atraso.

A Figura 6 mostra a taxa de descarregamento para diferentes valores de τ, bem como uma comparação entre os algoritmos *OppLite* e *Random* no cenário ROLLERNET. Como mostrado na Figura 6a, com $\Gamma_{relay} = 0$ o CaF apresenta uma taxa de descarregamento de (51, 88, 94, 95)% para $\tau = \{1, 100, 600, 1200\}$s, respectivamente. É importante ressaltar que a diferença nas taxas de descarregamento para o intervalo entre $\tau = 600$s e $\tau = 1200$s é menor ou igual a 1% $\forall \Gamma_{relay}$. Portanto, o *OppLite* quase alcança sua taxa



(a) Tolerância de atraso (*OppLite*) **(b)** *Random* vs *OppLite*

Figura 5. CaF: Taxa de descarregamento de dados no cenário INFOCOM



(a) Tolerância de atraso (*OppLite*) **(b)** *Random* vs *OppLite*

Figura 6. CaF: Taxa de descarregamento de dados no cenário ROLLERNET

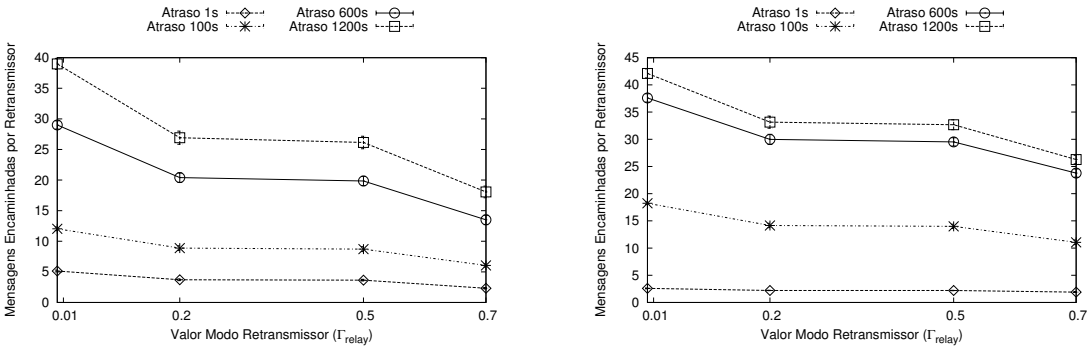
máxima de descarregamento com 10 minutos de tolerância de atraso.

A Figura 6b mostra a comparação de desempenho do CaF utilizando o o *OppLite* e o *Random*. A escolha de nós baseada em um conjunto de critérios, como realizado pelo *OppLite*, acarreta em um melhor desempenho do que a escolha de um conjunto de nós aleatórios, como realizado no *Random*, em especial para $\Gamma_{relay} \geq 0.5$.

De fato, o CaF com *OppLite* apresentou melhor desempenho em cenários com tolerância de atraso reduzida, devido às características do ROLLERNET. Neste *trace*, o *Random* seleciona nós retransmissores que raramente encontrarão nós oportunísticos.

A Figura 7 mostra a quantidade de mensagens encaminhadas por nós retransmissores nos cenários INFOCOM e ROLLERNET. No INFOCOM, Figura 7a cada nó retransmissor replicou de 4 ($\tau = 1$s) a 30 ($\tau = 1200$s) mensagens distintas em média. Já no *trace* ROLLERNET, cada nó encaminha de 2 a 35 mensagens distintas, $\tau = 1$ e $\tau = 1200$ (Figura 7b). A base de dados ROLLERNET provê mais oportunidades de contato e, como consequência, os nós retransmissores podem encaminhar suas mensagens com mais frequência do que no cenário com a base de dados INFOCOM.

Em ambos os cenários, o número de mensagens encaminhadas para todo valor de Γ_{relay} é quase constante quando a tolerância de atraso é reduzida. Neste caso, as replicações de uma mensagem são definidas pelo tamanho dos componentes conectados



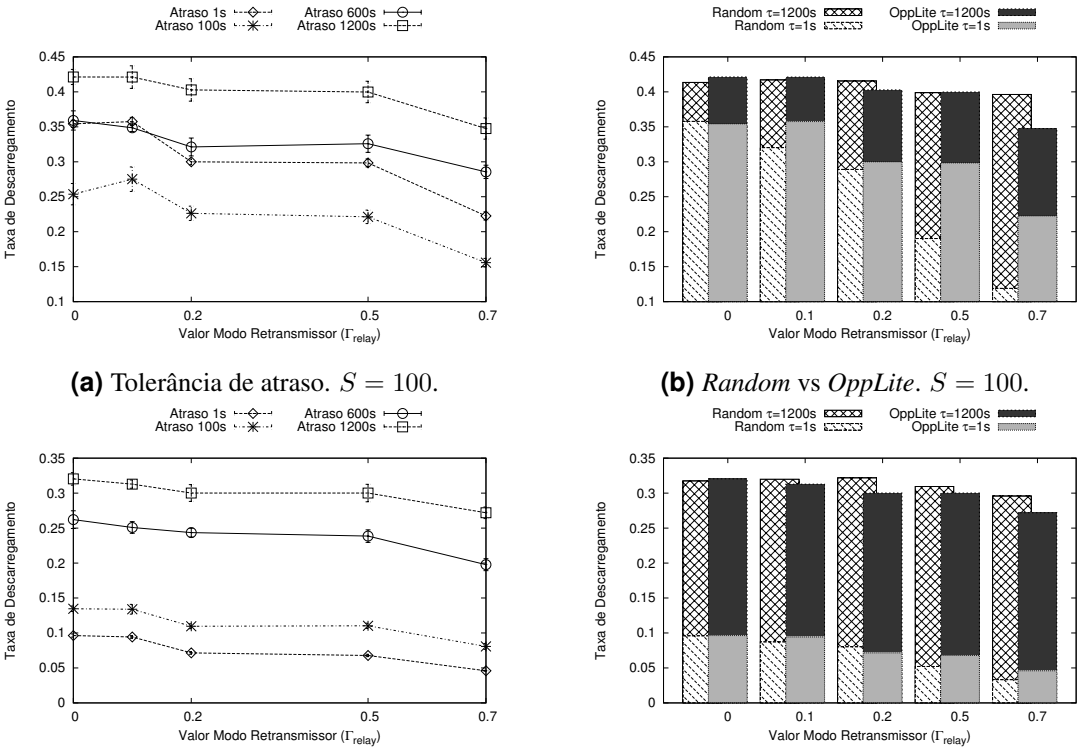
(a) INFOCOM **(b)** ROLLERNET

Figura 7. CaF**: Média de mensagens encaminhadas por nós retransmissores.**

ao qual o nó retransmissor pertence. Já valores elevados de τ permitem que os nós repliquem mensagens por mais tempo.

5.3. Análise do RaC

O RaC representa uma aplicação de requisição HTTP, na qual todos os nós requisitam conteúdo. Entretanto, a quantidade de conteúdo disponível para requisição, aqui referenciada como *pool size* (S), foi limitada em $S = [100, 10000]$ conteúdos distintos. Cada conteúdo tem uma chave de identificação e os nós escolhem uma chave seguindo uma distribuição Zipf, com isso, algumas chaves são requisitadas com maior frequência.



(a) Tolerância de atraso. $S = 100$. **(b)** *Random* vs *OppLite*. $S = 100$.

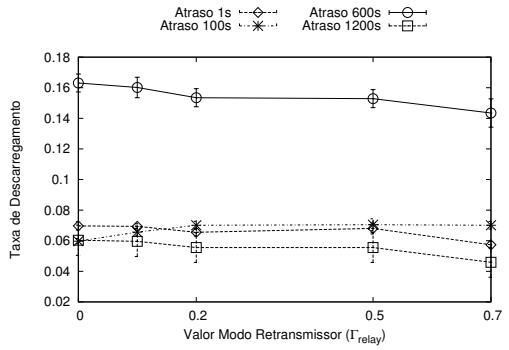
(c) Tolerância de atraso. $S = 10000$. **(d)** *Random* vs *OppLite*. $S = 10000$.

Figura 8. RaC**: Taxa de descarregamento de dados no cenário INFOCOM.**

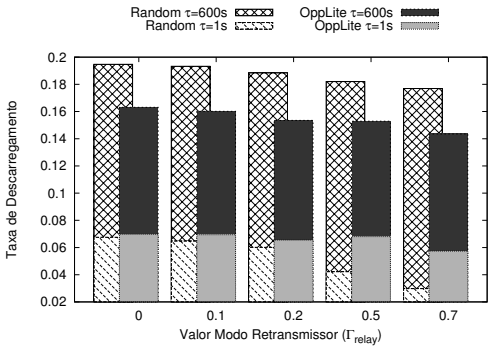
No RaC, a tolerância de atraso (τ) significa quanto tempo um nó oportunista aguarda até encontrar o conteúdo desejado no *cache* de um nó retransmissor. Após essa tolerância de atraso, os nós oportunistas requisitam o conteúdo utilizando a infraestrutura. Além disso, os nós oportunistas requisitam apenas um conteúdo até obter uma resposta, oportunisticamente ou diretamente da infraestrutura.

Os resultados no *trace* INFOCOM são mostrados na Figura 8. Quando há somente 100 conteúdos disponíveis para baixar (Figura 8a), RaC apresenta uma taxa de descarregamento de até 42% do tráfego se nós oportunistas suportarem 1.200 segundos de tolerância de atraso e $\Gamma_{relay} = 0$ ou 34% se $\Gamma_{relay} = 0.7$. Neste cenário, o RaC apresenta uma taxa de descarregamento de 22% a 35% do tráfego quando $\tau = 1s$, pois quanto menor a tolerância ao atraso, maior a probabilidade de nós vizinhos possuírem o conteúdo. A Figura 8b mostra um comparativo do RaC utilizando os algoritmos *Random* e *OppLite* com tolerância de atraso de 1 e 1200 segundos. O RaC com *OppLite* supera a solução com *Random* em 1 segundo de tolerância quando $\Gamma_{relay} \geq 0.1$.

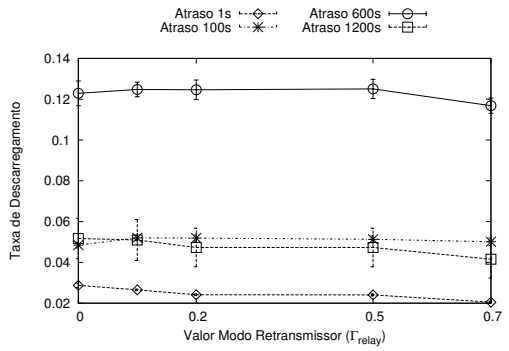
Quando a lista de conteúdo aumenta, a taxa de descarregamento decai, como pode ser visto nas figuras 8c-d. O algoritmo *Random* mantém a taxa de descarregamento de dados quase constante quando $\tau = 1200$s, devido ao número elevado de nós selecionados como retransmissores. O RaC com *OppLite* atinge uma taxa de descarregamento maior quando os nós oportunistas possuem baixa tolerância de atraso.



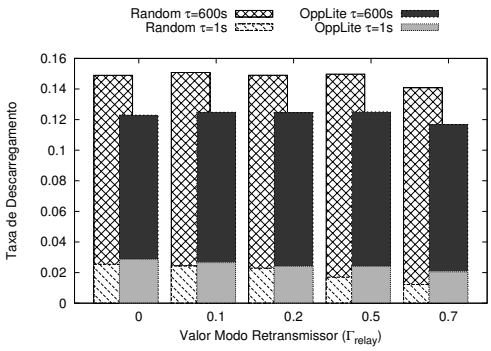
(a) Tolerância de atraso. $S = 100$.



(b) *Random* vs *OppLite*. $S = 100$.



(c) Tolerância de atraso. $S = 10000$.



(d) *Random* vs *OppLite*. $S = 10000$.

Figura 9. RaC: Taxa de descarregamento de dados no cenário ROLLERNET.

A Figura 9 mostra as taxas de descarregamento avaliadas no ROLLERNET. A melhor taxa de descarregamento foi obtida com uma tolerância de atraso de 600 segundos. Com $\tau = 600$s, o RaC apresenta uma taxa de descarregamento de até 17% do tráfego de

dados da rede no melhor caso (Figura 9a) e 12% no pior caso (Figura 9c). Neste cenário, as baixas taxas de descarregamento obtidas por $\tau = 100$s e $\tau = 1200s$ são devidos à duração da simulação e às características do *trace*. Como mencionado, um nó oportunista faz somente uma requisição por vez, tolerar $\tau = 1200s$ faz com que os nós aguardem muito tempo (40% do tempo simulado) antes de requisitar novos conteúdos.

O RaC com *OppLite* apresentou taxas de descarregamento 10% menores do que com *Random* com $\tau = 600$s no *trace* ROLLERNET, Figuras 9b e 9d, e igual ou maior do que *Random* quando $\tau = 1$s. Como mencionado anteriormente, o algoritmo *Random* seleciona mais nós do que o *OppLite* no *trace* ROLLERNET.

Apesar do *trace* ROLLERNET conter 1112 nós, apenas 62 nós estão ativos durante todo tempo de vida da rede, o que diminuiu a taxa de descarregamento quando comparado com o *trace* INFOCOM.

6. Conclusões

Neste artigo, apresentamos e discutimos três aplicações para avaliar o descarregamento de dados móveis utilizando o paradigma de comunicação dispositivo para dispositivo: (i) *Opportunistic Relaying* (*OpR*), nós oportunistas transmitem suas mensagens para a infra-estrutura através de nós intermediários; (ii) *Cache-and-Forward* (*CaF*), os nós retransmissores encaminham todo o conteúdo que recebem para os nós oportunistas; e (iii) *Relay as Cache* (*RaF*), os nós oportunistas procuram por conteúdo no *cache* dos nós retransmissores.

Na aplicação *OpR*, observamos que os nós retransmissores podem transmitir até 35% do tráfego em um cenário de conferência com um atraso de menos de 20 minutos. Em um cenário representando uma multidão em movimento nas ruas de Paris, os nós retransmissores podem transmitir até 70% do tráfego. A aplicação *CaF* descarrega entre 30% e 80% do tráfego de dados no INFOCOM e entre 52% e 94% do tráfego de dados no ROLLERNET com um atraso de 20 minutos. A aplicação *RaC* descarrega entre 32% e 44% do tráfego de dados quando os nós oportunistas toleram 20 minutos de atraso no INFOCOM. Enquanto isso, no *trace* ROLLERNET, o RaC descarrega entre 7% a 17% com 10 minutos de atraso.

Essas avaliações mostraram o potencial do uso de dispositivos pessoais como *cache* para descarregar o tráfego de dados de celular para rede oportunista com um atraso pequeno. Observamos que há um compromisso entre o número de nós retransmissores e a quantidade de mensagens descarregadas: quanto maior o número de nós retransmissores, mais alta é a taxa de descarregamento de dados. Contudo, isso requer a disposição dos usuários para mudar seu dispositivo para o modo de retransmissão.

Além disso notamos que, para todos os cenários e aplicativos, o atraso de entrega aceitável de uma aplicação pode limitar a quantidade de mensagens descarregadas. Por exemplo, aplicações de sensoriamento participativo suportam 20 minutos de atraso antes de enviar os dados. Por outro lado, os usuários podem necessitar de atraso inferior para enviar um e-mail.

Estas descobertas podem ajudar a comunidade de pesquisa a focar o desenvolvimento de algoritmos que tentam maximizar a taxa de descarregamento enquanto minimizam o atraso dos dispositivos no modo oportunista.

Referências

3GPPP (2013). Study on architecture enhancements to support proximity services (prose) (release 12). Report 23.703 V0.4.1, 3rd generation partnership project.

Barbera, M. V., Stefa, J., Viana, A. C., De Amorim, M. D., and Boc, M. (2011). Vip delegation: Enabling vips to offload data in wireless social mobile networks. In *Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–8. IEEE.

Choi, Y., Yoon, C.-h., Kim, Y.-s., Heo, S. W., Silvester, J., et al. (2014). The impact of application signaling traffic on public land mobile networks. *Communications Magazine, IEEE*, 52(1):166–172.

Cisco (2015). Global mobile data traffic forecast update, 2014-2019.

Doppler, K., Rinne, M., Wijting, C., Ribeiro, C., and Hugl, K. (2009). Device-to-device communication as an underlay to lte-advanced networks. *Communications Magazine, IEEE*, 47(12):42–49.

Haggle (2013). *Haggle Project*. Haggle. http://www.haggleproject.org.

Haldar, K. L., Li, H., and Agrawal, D. P. (2013). A cluster-aware soft frequency reuse scheme for inter-cell interference mitigation in lte based femtocell networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6. IEEE.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The one simulator for dtn protocol evaluation. In *Conference on Simulation Tools and Techniques*, pages 1–10.

Mota, V. F., Macedo, D. F., Ghamri-Doudanez, Y., and Nogueira, J. (2014). Managing the decision-making process for opportunistic mobile data offloading. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–8. IEEE.

Mota, V. F. S., Macedo, D. F., Ghamri-Doudane, Y., and Nogueira, J. M. S. (2013). On the feasibility of wifi offloading in urban areas: The paris case study. In *IFIP Wireless Days conference*.

Paul, S., Yates, R., Raychaudhuri, D., and Kurose, J. (2008). The cache-and-forward network architecture for efficient mobile content delivery services in the future internet. In *Innovations in NGN: Future Network and Services*, pages 367–374. IEEE.

Rebecchi, F., Dias de Amorim, M., Conan, V., Passarella, A., Bruno, R., and Conti, M. (2015). Data offloading techniques in cellular networks: A survey. *Communications Surveys Tutorials, IEEE*, 17(2):580–603.

Tournoux, P., Leguay, J., Benbadis, F., Conan, V., De Amorim, M., and Whitbeck, J. (2009). The accordion phenomenon: Analysis, characterization, and impact on dtn routing. In *IEEE INFOCOM*, pages 1116–1124.

Valerio, L., Abdesslemy, F. B., Lindgreny, A., Bruno, R., Passarella, A., and Luoto, M. (2015). Offloading cellular traffic with opportunistic networks: a feasibility study. In *14th Mediterranean Ad Hoc Networking Workshop*, pages 1–8. IEEE.

Whitbeck, J., Lopez, Y., Leguay, J., Conan, V., and Amorim, M. (2012). Push-and-track: Saving infrastructure bandwidth through opportunistic forwarding. *Pervasive and Mobile Computing*, 8(5):682 – 697.

Trilha Principal do SBRC 2016
Sessão Técnica 25
Computação Móvel e Ubíqua

Controlling Swarms of Unmanned Aerial Vehicles using Smartphones and Mobile Networks; an evaluation of the Latency requirements

Bruno Olivieri[1], **Marcos Roriz Junior**[1], **Markus Endler**[1]

[1]Department of Informatics – Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro – RJ – Brazil

{bolivieri,mroriz,endler}@inf.puc-rio.br

Abstract. *Swarms of Unmanned Aerial Vehicles (UAV) can be useful to several applications, such as urban surveillance and monitoring of mass events. To build and maintain a swarm formation, the UAVs need to compute, communicate, and coordinate their actions. However, it is complex to design and integrate computing and communicating chips into the UAV control board. Thus, in this paper, we argue that out-of-the-box Smartphones can be coupled to UAV and fulfill this role as a proxy computing unit, i.e., they can be attached to UAVs to do their computing and communicating tasks. Based on this hypothesis, we present a framework to coordinate the UAV swarms using conventional mobile networks. Then, we evaluate if this approach, under different latencies and swarm speed, is fast enough to coordinate the UAV Swarms. The results indicate that the proposed approach present stable swarm formations under* $200\,ms$*, a higher limit than standard 3G and 4G latencies. Further, the results show that the ratio* $\frac{\textit{leader speed}}{\textit{communication latency}}$ *is the key aspect that determines the swarm formation accuracy.*

1. Introduction

Unmanned **A**erial **V**ehicles (**UAV**) are useful in numerous application scenarios, specifically those unreachable or dangerous to humans, such as area patrol, disaster rescue missions, and surveillance monitoring of massive events. They are useful for these scenarios because the aircraft can be remotely operated by a computer through a radio-link, rather than by a human pilot [Austin 2010]. Such UAV's are common referred as flying robots or drones, due to the fact that their computer system are capable of a certain level of self-control, *e.g.*, run arbritary code, move to a given direction, or fly in a given altitude.

In general, one needs several UAVs to perform these tasks, for example, to cover a larger area, or to get multiple views of a given site. A set of UAV that work together to execute a given task is called a *swarm*. Swarm members work together by communicating their position and other useful information in predefined intervals. Spatial formation is a key element for swarm collaboration, *i.e.*, how the UAV are disposed in space (*e.g.*, circular, rectangular, flock, *etc*).

To enable such coordination, UAV swarm members need to communicate with each other. The communication infrastructure used by UAVs can be based on non-existing or existing network infrastructures, such as dedicated backbones or privated repeaters. In the first case, the swarm itself establishes and maintains an ad-hoc communication network to each other through a radio signal, while in the latter case the UAV communicate to

each other through an ubiquitous and uniform wireless communication medium, such as mobile networks (*e.g.*, 3G, 4G). Although non-existing communication network may be mandatory for certain applications, especially those in non urban locations, the existing network infrastructure is a natural fit for urban environments, where one can take advantage of the wide-range coverage of mobile cellular networks, as this works relies.

Nonetheless, it can be complex to design and integrate computing and communicating chips with the UAV control board. Thus, we argue that out-of-the-box Smartphones, *e.g.* Android and iPhone, can fulfill this role as a proxy computing unit, that is, they can be attached to the UAVs to do their computing and communicating tasks. Not only several UAVs are powerful enough to carry an out-of-the-box smartphone, these devices also present a rich set of sensors, such as location, altitude, accelerometer, *etc*, which can be used to enrich/complement the UAV's sensored data. Further, smartphones are constantly becoming lighter (in terms of weight) and cheaper to manufacture.

Using the communication infrastructure, several UAV swarm systems coordinate themselves based on the concept of **P**oint **o**f **I**nterest (POI), where the swarm formation must cover, patrol, or explore the set of POIs [Şahin and Spears 2005]. However, it is difficult to instrument an area with sensor or static routes, especially because POIs can dynamically change depending on the application context. A solution for this problem is to enable the UAV swarm to move towards POIs steered by an (human) operator, and if their sensing capabilities around a POI need to be improved or enlarged, then the swarm should move in a rigid formation.

However, movement coordination in UAV swarms is a challenging task, since it requires reliable communication and timely mutual sharing of the UAV's states, *e.g.*, to ensure a good coverage, avoid collision, *etc*. Further, if a specific swarm formation (shape) should be maintained, then it must be ensured that their current positions and other movement parameters are timely communicated, the robots are always kept within safe distance from each other so that potential collisions are avoided.

Concerned with the requirement of timely swarm coordination and communication, in this paper we investigate **if mobile networks (*e.g.*, 3G, 4G) are fast enough to coordinate Smartphone-based UAV applications**. To do that, by considering premises and limitations of the mobile network infrastructure, we built a coordination framework (master-slave) based on a mobile message-oriented middleware. We simulated an application instance of the proposed framework with several network latencies to validate the model. By examining the coordination accuracy compared to the different network latency values we suggest the range and scenarios where mobile networks can be viable for UAV swarm applications. This can be summarized in the following contributions:

- Answer the question: Is it possible to control a swarm of UAVs using mobile networks and smartphones? If positive, what are the primary constraints?
- Synthetic evaluation of the proposed approach using mobile networks to discover the latency requirements.

The remainder of the paper is structured as follows. Section 2 overviews the fundamental concepts used throughout the paper, while Section 3 presents the core concepts and algorithms of the proposed framework. In Section 4 we evaluate an instance (application) of the model using a serious of network latencies. Section 5 presents the related works. Finally, in Section 6 we present the conclusion and final remarks.

2. Fundamental Concepts

In this section we briefly review the different concepts used throughout the paper, which can be subdivided into those related to UAV systems, those related to the underlying communication infrastructure, and the ones related to the existing coordination models.

2.1. Unmanned Aerial Vehicles (UAV)

A UAV can be simply described as an aircraft which replaces the aircrew by a computer system and a radio-communication link. UAV can have different shapes, which will drastically vary conforming the intended application, as illustrated by Figure 1. For example, zeppelins are used for long and slow movement aerial surveillance, airplanes for long distances, and quadcopters for agile movement hovering over dense urban areas.

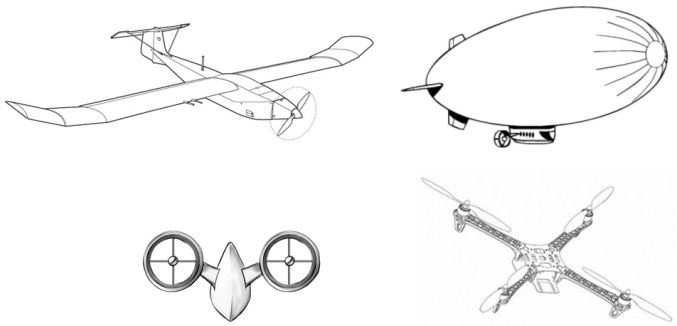


Figure 1. Diverse UAV models: starting with a fixed wing (top left); a Zeppelin (top-right); a bicopter (bottom-left) and a quadcopter (bottom-right).

Based on the existing literature, UAVs can be referred by many different names, such as Drones, Flying Robots or RPAS (remote piloted aircraft systems). In all cases, these definitions consider that the airship has an computing component, a control subsystem, and possibly a radio-link to comunicate with the ground units. Thus, flight is controlled either autonomously by onboard programs and/or through remote control by a pilot located on the ground. Despite this variety of UAV models and a wide range of possible applications, this work approaches UAVs mainly from a computing perspective, instead of its flying characteristic, by focusing on UAVs that are able of executing certain autonomous tasks based on simple computing primitives (operator commands) such as takeoff, landing and traveling to a Point of Interest (POI).

Precisely, in this paper we use a specific type of UAV, a quadcopter – a kind of multirotors (illustrated by Figure 1 bottom-right) – as it present some advantages for executing simple autonomous tasks based on computing commands. First, because it is not a fixed-wing aircraft, quadcopters are capable of easily hovering over a stationary POI. Thus, the device can hold its position replicating helicopter capabilities.

Second, a quadcopter has four propellers with simple motor connection without helicopters rotors. As a result, the aircraft is capable of moving around with four degrees of freedom, flying from a three dimension point $\langle latitude, longitude, height \rangle$ to another one. This feature by itself is very useful to implement simple autonomous tasks (the building blocks of coordination algorithms), without concern of the aerodynamics, since the position and directional vector describing UAV motion are independent of aerodynamic lift pressure (which is an essential factor for airplanes), or even temperature or winds (as is the case for Zepelins).

2.2. Communication Infrastructure

Regarding the communication infrastructure, as mentioned before, we propose to use the Internet access provided by mobile networks to establish communication between several UAVs, in order to enable and control the UAV system. The current mobile network technologies (2G/3G/4G) provide wide-range Internet connection through IP networks for mobile phones. Hence these phones may play the role of universal communication (and processing) hubs for the UAVs. This decision to use the mobile Internet brings some advantages and limitations, as noted below. The primary positive aspects are:

1. Reliance on a established set of wireless communication standards and protocols;
2. Ability to benefit from already installed and tested networks;
3. Wide (and growing) wireless coverage in metropolitan areas;
4. Continuous improvements and optimizations of the used protocols;
5. Potential to utilize a large set of smartphones as radios for communication.

The negative aspects of mobile networks for UAV are:

1. Higher and less predictable network latency compared with the latencies in other ad-hoc networks used for MANs as privated WLANs;
2. Higher package loss rates due to the common handover process between cellular base-stations providing the telecom data services;
3. Due to more frequent handovers caused by the high mobility of UAVs, it is harder to maintain continuous data streams in these networks.

To tackle the second and third negative issues presented, we base our work in existing mobile message-oriented middleware platforms, such as **S**calable **D**ata **D**istribution **L**ayer (SDDL) [David et al. 2013b]. These platforms present lightweight mobile protocols, such as RUDP and MR-UDP [David et al. 2013a], that address intermittent connections, including soft handover. Such protocols can also maintain data-streams to the UAVs through the usage of low-rate keep-alive (heartbeat) messages.

Further, the event-based programming models provided by message-oriented middleware, *e.g.*, *publish/subscribe*, seems a natural fit for the distributed nature of UAV systems. A primary advantage of this programming model is the asynchronous and topic-based messaging primitives, enabling decoupled elements (the UAVs) to easily share state between them by associating a pub/sub topic to the desired event or coordination message type. For example, an UAV leader can easily share (publish) its state with the remainder UAVs in the swarm using an 1-to-N subscription topic, where the swarm is subscribed to the UAV leader topic. By choosing this programming model, we are able to reduce the number of messages, ease the distributed UAV coordination using topics to represent the events, and do "groupcast" messages.

2.2.1. Coordination Model

During the last three decades, multiple coordination models for robots (including UAVs) have been proposed [Murray 2007]. This growing interest in these models was motivated by the premise that certain tasks could be better performed by employing multiple simple robots rather a single (or a few) powerful robot [Pal et al. 2005].

Severe models (e.g. [Prencipe 2005]) follow a distributed coordination approach, where the swarm state is distributed, and the members communicate the state only through message passing. In this work, we opted for using a semi-synchronous model and the coordination life cycle can be summarized as follows:

- **Wai**t: A not synchronized phase, when a UAV waits for information of the leader's new location;
- **Look**: A synchronous phase that starts when a UAV sense its local environment, *e.g.*, its own position and contextual situation (such as battery level or actual direction or status), and is able to use this data to take decisions within the scope of the algorithm that control its behavior.
- **Compute**: Using the sensed and external information, the UAV logically execute the coordination algorithm, and possibly move to a new state. For example, using the sensed location, the UAV identify that it is now in the "outside formation" state and it should retrieve to a given point.
- **Move**: Physically move/execute a given task, such as move to a given latitude and longitude, accelerate, etc.

3. Coordination Framework

In this section we describe the coordination framework for coordinating UAV swarm, which is based on three pillars: an UAV Smartphone-centric hardware architecture, the mobile message-oriented middleware publish/subscribe programming model, and a distributed coordination algorithm. We discuss and explain these elements throughout this section.

3.1. System Premises

In order to fully understand the coordination framework proposed in this work and its limitation, it is necessary to outline the assumptions made, as well as the main hypotheses related to the characteristics of the UAVs, the communication links and the location technology. The premises is that each UAV:

- Contains a location sensor (such as GPS, digital compass, etc);
- Is able to carry a Smartphone with the required sensors;
- Do not fail or deplete their batteries during operation.
- Has a reliable wired connection with the carried Smartphone, and the communication latency between the smartphone and the control board is negligible;
- UAV' control Board provides and reacts reliably to steering controls through an API, such as `MoveToPosition()`, `IncreaseSpeed()`, `DecreaseSpeed()`, `GoToAltitude()`, etc. The interaction with the control board is logically made by the Smartphone and executed analogously by the Flight Control Board;
- Has access to a mobile network through the entire geographic area of interest.

The framework uses a role-based concept (master and slave) and it is leader-oriented, in a sense that the UAVs steers the motion of the entire swarm around the leader in an orderly manner. Once a leader is appointed (master) by the user, the remaining UAVs are queried and some are designated as slaves of this leader UAV and thus gather around it. In other words, it is the user's responsibility to determine the number of slaves and the leader UAV. However, the algorithm is the one responsible for "recruiting" the designated number of slaves.

The swarm is said to be properly formed when the desired number of slaves surround the leader and assume their expected relative positions. This arrangement focuses on positioning the leader in the middle of a circle before arranging the slaves around it, thus increasing the covered area. The user, acting as an overseer, can, just by controlling the leader, direct several UAVs over a POI, thus increasing the visibility in that area.

3.2. Hardware Integration

A central aspect of this paper is the Smartphone-based UAV architecture, where the smartphone acts as proxy computing and communicating unit for the UAV control board. The idea is to decouple the architecture in two core parts, taking advantage of the processing and communicating power of smartphone devices and the well-tested UAV flight control boards. With respect to the airship functionality, the implicit assumption is that it has sufficient energy to perform the tasks, *i.e.*, perform the takeoff and landing, reach the swarm, and do the swarm flight task area autonomously.

An overview of the hardware architecture is illustrated in Figure 2, where the smartphone send and receive commands to the UAV control board. The smartphone receives commands from the coordination framework, such as go forward or goes for a coordinate. This solution is better described on a previous work [Olivieri and Endler 2014]

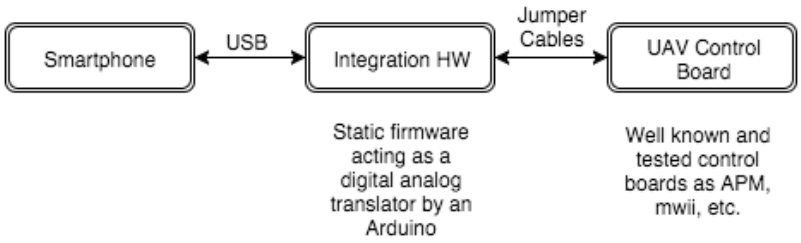


Figure 2. An overview of the UAV Smartphone-based hardware architecture.

3.3. Roles and Distributed Algorithm

To coordinate the swarm, we propose a distributed algorithm, that is, each UAV process the incoming commands itself without requiring a central processing node. Commonly, the elements in a distributed algorithm execute the same code, thus, we assign roles to distinguish them. First, we categorize UAV in two flying modes: Patrol and Swarm. In Patrol mode, the UAVs is autonomously patrolling the region within the borders of interest or executing user commands, while in Swarm mode, the UAV is aiding other UAV in a given task.

The Swarm mode can be further subdivided in two other ones: Leader and Slave. The user interact directly with the Leader UAV by sending control commands, such as move to a position, speed up, slow down, *etc*. Once the command is done, the leader UAV will hover and propagates its own coordinates to its slaves via groupcast (1-to-N message topics). Slave UAVs aid the leader one by responding to its commands, for example, to calculate and reach their expected position in the swarm.

These roles are a central part of the publish/subscribe communication platform. Each role is assigned to a different topic, precisely:

- $Patrol$: to communicate with patrolling (free) UAVs;
- $Swarm$: swarm slaves subscribe to this topic using the leader ID as filter.

Aside from the roles topics, we consider a generic topic (UAV) that every aircraft is subscribed to, but using their ID as filter. This is useful to communicate directly ("unicast") with the given UAV, for example, an user operator sending commands to a given UAV.

Whenever the operator needs that a particular UAV should be escorted by a swarm, *e.g.*, a relevant event occurred, it elect that UAV as a swarm leader. In addition, it also indicates the number n of additional UAVs (from the total of m devices) that will be requested to establish the swarm (where $n < m$). At this moment, all the UAVs subscribed to the $Patrol$ topic receive a message request to send (publish) their current position coordinates to the UAV designated as leader. Algorithm 1 and 2 presents the UAV main loop and the reaction routines for each role topic.

When the leader receives the first slave candidates' messages, it waits for a time interval equal to $2 \times \Delta$ to receive the remaining or late slave candidates messages. Once in possession of this data, the leader selects the n "most appropriate" UAVs that are to become part of its swarm. The criteria used for selecting the slave UAVs vary, depending on the task: it might be the distance to the leader, the airship's residual energy, or any combination of these or other airship data. After the selection is done, the leader UAV publish a command to the elected swarm. When the selected members receives this message, they immediately switch to the slave role, unsubscribe from the $Patrol$ topic, and subscribe it to the leader location updates in the $Swarm$ topic. Consequently, any location update message sent by the leader to this topic is automatically "groupcast" to the slave UAVs subscribed to the topic.

The swarm formation will assume a circular form around the leader, with radius r, in order to widen the visual coverage of the detected occurrence at the ground. All n slave UAVs will be positioned on this circle $\theta = \frac{360}{n}$ degrees apart from each other. To sustain this formation, all slave UAVs receive at the start of the swarm a leader message containing a θ angle and their respective relative positions on this circle (clockwise) around the leader's position (l_x, l_y). To exemplify, the first slave will be positioned at $(r \times \cos(\theta), r \times \sin(\theta))$, while the 2^{nd} slave will be placed at $(r \times \cos(2\theta), r \times \sin(2\theta))$, and the i-th slave will position itself at $(r \times \cos(i\theta), r \times \sin(i\theta))$, as illustrated by Figure 3.

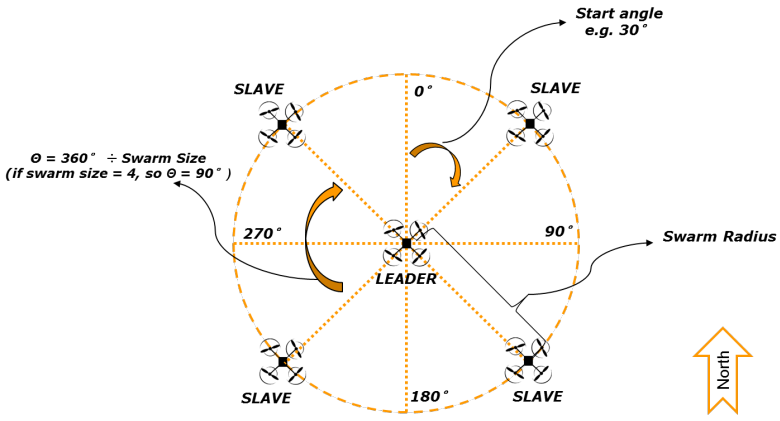


Figure 3. Swarm sample describing its relative positions and main formation.

Algorithm 1: UAV Reactive Loop

```
1  on initialization(uav_id):
2      id ← uav_id
3      state ← Patrol
4      subscribe to Patrol and UAV topic
5  on receive message msg from UAV topic:
6      switch msg_type do
7          case Elected as Leader by Operator
8              BECOMELEADERROUTINE (msg)
9          case Slave Response
10             HANDLESLAVERESPONSE (msg)   ▷ Slave Candidates Response
11         case Release Command
12             if status = Leader then
13                 publish Release Command to each slaves (UAV topic)
14             status ← Patrol
15             leader ← null
16             angle ← null
17             speed ← speed × 0.6
18             unsubscribe from Swarm topic
19             subscribe to the Patrol topic
20 on receive message msg from Patrol topic:
21     switch messagetype do
22         case Slave Request
23             HANDLESLAVEREQUEST (msg)
24         case Slave Confirmation
25             HANDLESLAVERESPONSE (msg)
26 on receive message msg from a leader of the Swarm topic:
27     p_x ← r. cos(angle)
28     p_y ← r. sin(angle)
29     destination ← p ⊕ msg_position        ▷ ⊕ = coordinates (x, y) addition
30     move to destination
31 main loop:
32     while true do
33         if status = Patrol then
34             hoverAround()                 ▷ Stays patrolling
35         if status = Leader then
36             wait 2 × Δ                    ▷ Disseminate leader position
37             publish new position to Swarm topic
```

Algorithm 2: UAV procedures

procedure BECOMELEADERROUTINE *(msg)*:
  $status \leftarrow Leader$
  $swarmsize \leftarrow msg_{swarmsize}$
  $swarmcandidates \leftarrow \emptyset$
  publish to $Patrol$ topic a $SlaveRequest$ message from id
  wait for $2 \times \Delta$ for slave responses
  $slaves \leftarrow$ **choose** $swarmsize$ candidates from $swarmcandidates$
  for $i \leftarrow 1$ *to* $swarmsize$ **do**
    $cmsg \leftarrow$ create a new $SlaveConfirmation$ message
    $cmsg_{leader} \leftarrow id$
    $cmsg_{angle} \leftarrow (\frac{360}{swarmsize} \times i\text{++}) + 30$
    publish cms to $Patrol$ topic using c id

procedure HANDLESLAVEREQUEST *(msg)*:
  publish to UAV topic of msg_{id} a $SlaveResponse$ message with $\langle id, pos \rangle$

procedure HANDLESLAVERESPONSE *(msg)*:
  $swarmcandidates \leftarrow swarmcandidates \cup \langle msg_{id}, msg_{pos} \rangle$

procedure HANDLESLAVECONFIRMATION *(msg)*:
  unsubscribe from $Patrol$ topic
  subscribe to $Swarm$ topic of leader msg_{id}
  $status \leftarrow Slave$
  $leader \leftarrow msg_{id}$
  $angle \leftarrow msg_{angle}$

All slave UAVs conforms to this configurations, which remain unchanged during the swarm, and use it to move to their relative position (in the circular formation) when the leader moves. The θ angle is measured from the same referential, the geographical north, which is known by each UAV through their compass and is referred to as mutual North. From this moment on, the leader will continuously transmit its absolute position (l_x, l_y) to all its slaves through the *Swarm* topic. This position-update message will be received by all slave UAVs within the $2 \times \Delta$ time limit, and will allow them to constantly update their absolute positions and keep the swarm formation.

The algorithm will continue the logic until the user operator send a release message to the leader and the swarm, which instruct all UAVs in the swarm to return to $Patrol$ mode. Once they change their states, all UAVs will resume their task of ordinary patrolling, which they perform autonomously.

3.4. Algorithm Complexity

The coordination algorithm proposed here is a distributed algorithm with low message complexity, both in the phase of swarm formation and during the swarm mode flight. More specifically, as soon as a new leader is determined with n slaves, it will publish a *Slave Request* to all *Patrol* UAVs ($\leq m$), some of which will reply, yielding a maximum of

$2 \times (m-1)$ messages. Next, the leader sends out unicast messages (*Slave Confirmation*) for the n selected slaves (resulting in n messages), which determine the members of the swarm. A swarm formation is considered acting correctly when its slaves were recruited and are following its leaders as well. To ensure that the chosen UAVs are flying in swarm formation, the leader only needs to send publish its location in the *Swarm* topic every time it changes its position, as each of its slave UAVs will be able to recalculate its own absolute position. Thus, each slave recruitment process and swarm update requires only a small and predictable number of messages, $\mathcal{O}(m)$ and $\mathcal{O}(n)$, respectively, where m is the total number of UAV and n is the number of UAV in a swarm ($n < m$).

4. Evaluation

To evaluate the feasibility of the proposed approach, we evaluated an application instance of the coordination framework with different network latencies and swarm speeds[1]. This evaluation allowed us to identify some issues and constraints of the proposed approach, including an insight of the latency requirements and swarm speed, specifically if fast swarms are able to be controlled in the presence of mobile network latency.

4.1. Evaluation Setup

To evaluate our approach we opted for simulating several UAVs using parallel process communicating with each other. Workarounds were implemented in order to emulate the two main problems that might have impact the scenario: network latency and eventual message loss. These issues were artificially inserted in the simulation environment to test their respective consequences on the accuracy of the swarm formation.

The mobile network latency was emulated by using an internal variable (Δt) in the message-sending functions. Thus, each time a UAV sends a message, rather than being processed immediately, it is buffered into the mobile protocol queue. The elements in this queue are processed with a delay of Δt time period. For example, assuming an average latency of 50 ms for 4G networks, each time a UAV sends a message, it takes 50 ms before it is actually transmitted. So, a message takes $2 \times \Delta t$ from UAV A to UAV B, one Δt from UAV A to the middleware and another Δt to reroute from the middleware to B.

Message loss is the second network behavior that was emulated, and it was approached in a similar way as the introduction of network latency. Another variable (ϕ) hold the probability (in percentage) of message-delivery success. Thus, each time a UAV tries to send a message, the simulator uses the ψ value to determine whether the message will or not be actually transmitted. For instance, for $\psi = 90\,\%$, every time an UAV send a message there is $90\,\%$ chance of the message being transmitted.

We also simulated the UAV "control board", which includes function to alter the UAV movement, altitude, direction, etc. These attributes are continuously updated by a thread that describes the airship's movement physics and processes external flight control commands received through the coordination framework. It also includes high-level routines, such as `GoToPoint()`, `IncreaseSpeed()`, `ChangeAltitude()`.

To verify if the swarm spatial formation is correct we implement a server side element that subscribes the leader and swarm location updates, relying on a Complex Event

[1] An on-line video describing the experiment is available at `https://www.youtube.com/watch?v=3phH-5e7l9c`

Processing engine [EsperTech 2015] framework. This element is useful since it enables us to verify how the swarm formation change with the experiments, for example, when a leader change its position, if and the elapsed the spatial formation becomes invalid.

Finally, all experiments were performed in the same environment, consisting of a single machine running the Ubuntu GNU/Linux 14.04 operating system, and the Java distribution OpenJDK 7. The machine has an Intel i7 quad-core (with Hyperthread Technology, enabling the execution of up to eight processes simultaneously), and 16 GB of RAM memory. The coordination framework was implemented in Java, and used the Scalable Data Distribution Layer (SDDL) mobile message-oriented middleware.

4.2. Swarm Delay

The simulation scenario is composed by a UAV leader that is escorted by four slaves, equally distributed around a radius of 30 meters, that means, the slaves are positioned at the following angles 30°, 120°, 210°, and 300°. To evaluate how the network latency affects the swarm formation, we emulated eight network latencies and defined four different UAV speeds for the leader, while fixing the slave UAV speed to $120\,\%$ of the leader one.

In this test, in addition to using the coordination framework topics, all UAVs publish the location pair $\langle current, target \rangle$ to a ***Measurement*** topic. Messages sent to this topic are delivered without delay to the measurement server, ***i.e.***, without any artificial transmission delay, while messages in ordinary topics are artificially delayed by Δt period, as explained in subsection 4.1. We considered eight different Δt delays: $0\,ms$, $50\,ms$, $100\,ms$, $150\,ms$, $200\,ms$, $210\,ms$, $225\,ms$, and $250\,ms$. For each latency, we also considered different speeds for the leader UAV, $2\,m/s$, $5\,m/s$, $10\,m/s$, $15\,m/s$.

The measurement of the swarm formation accuracy starts when the UAV swarm is first formed. From this point on, the measurement tool computes the swarm's formation accuracy each time it receives the location pair $\langle current, target \rangle$ of an UAV from the *Measurement* topic. By considering the difference between current and target location of an UAV, and the other UAVs records, we are able to detect if a swarm formation is accurate. To handle minor imprecision, we consider a formation valid if all UAV slaves are within either $90\,\%$ to $110\,\%$ of the swarm's R radius as more discussed on [Olivieri and Endler 2015]. Each test run analyzes $10\,000$ messages from each UAV, in addition, each test were executed four times, resulting in $10\,000 \times 5 \times 4 = 200\,000$ messages being analyzed.

The simulation results are illustrated in Figure 4, which presents the relationship between latency and swarm correctness, for speeds of 2, 5, 10, and 15 meters per second. Emulated latencies were increased by $50\,ms$. The vertical bars show the standard deviation range of each test suite. The graph shows that the best case scenario are those where the UAV is moving slower ($2\,m/s$) with a lower latency ensuring 100% of correctness, while the worst-case scenario happens with a combination of fast speed ($15\,m/s$) and higher latency, cause higher imprecision in the swarm formation with only 40% of correctness. Further, slower swarms yield better formation, since delayed messages will impact them less than faster ones. For example, swarms moving at $2\,m/s$ could present perfect ($100\,\%$) formation, even at the presence of $200\,ms$ latencies. On the other hand, swarms moving at $5\,m/s$, presented correct formation on only $80\,\%$ of the cases. This lead to the conclusion that the $\frac{\text{leader speed}}{\text{communication latency}}$ ratio is the key aspect that determines the swarm formation accuracy.

The experimental results showed that all series exhibited a non-linear and non-proportional shape. Interesting, for all speeds tested, the results shows that one can identify a maximum latency that is supported, after which the swarm formation's accuracy decreases significantly. These limits appear somewhat close to each other on the graph, indicating the presence of other factors potentially contributing to this behavior as discussed below.

In the scope of the present analysis, the results indicate at least another parameter that influences the swarm formation accuracy, the swarms' radius R, *i.e.*, the distance between the leader and the slave UAVs in the swarm. In order to evaluate if and how the swarm's formation accuracy depends on R, we repeated the previous experiment, but using $R = 50$ meters. The graph illustrated by Figure 5 shows the experiment result.

When comparing this graph with the one of Figure 4 it is clear that, when considering a larger radius, the overall swarm formation accuracy increases for higher speeds. On the other hand, increasing the swarm's radius does not result in a better network latency tolerance. Similarly, the significant decrease in the swarm formation's accuracy that was observed at $200\,ms$ did not change with the wider swarm radius. Nonetheless, when the swarm radius increases, all network latencies resulted in higher accuracy levels.

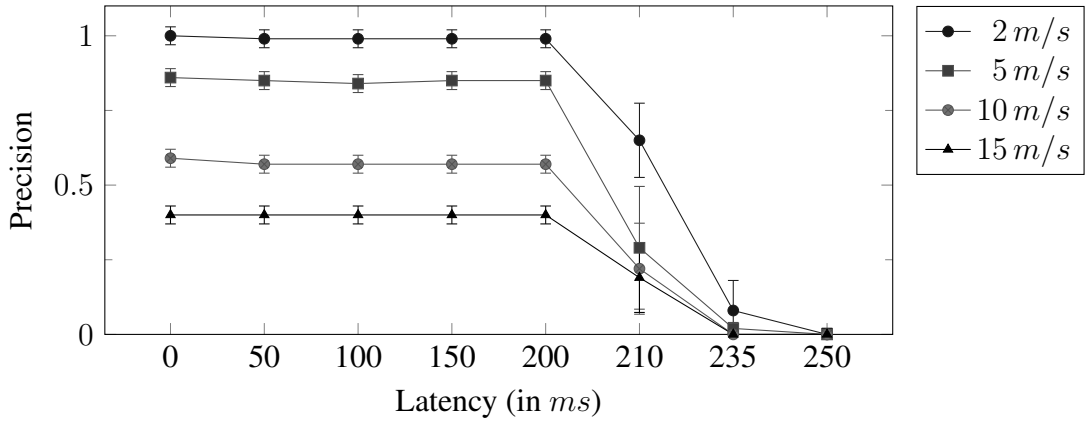


Figure 4. Graph showing results of different leader speeds and network latencies and their combined impact on the accuracy for the swarm formation.

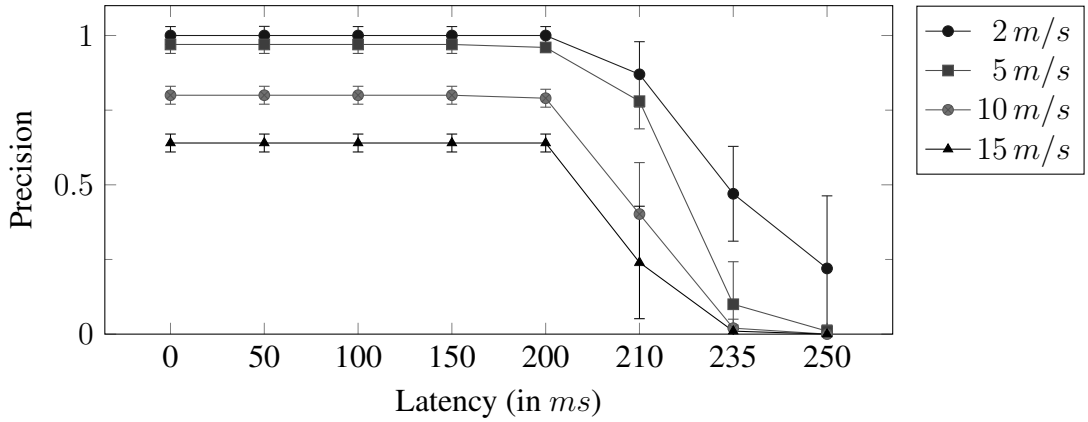


Figure 5. Same experiment of Figure 4, but with swarm radius equal to 50 meters instead 30 meters.

This improvement is due to the higher tolerance for the swarm accuracy. As previously noted, when the swarm radius increases, it make easier for all slaves to reach their target location. Further, the radius tolerance, here $10\,\%$, is also an important variable to be taken into consideration in the experiment. Nevertheless, this evaluation does not intend to explore all possible factors that can affect the swarm formation accuracy. However, we identified the three factors that seem to be more influential on the swarm's behavior, at least in a simulation environment, are: swarm speed, swarm radius, and network latency.

5. Related Work

Flying POIs surveillance using UAV is an active research area and usually been theoretically investigated under TSP and VRP problems optimizations [Pascarella et al. 2013]. Darrah *et al.* in their work [Darrah et al. 2015] introduces a Genetic Algorithm to perform a faster TSP like approach to use UAV on surveillance POIs problem but does not discuss the communication aspect. Barton *et al.* [Barton and Kingston 2013] focus on rapidly creating the swarm formation. They consider a V2I communication infrastructure and a machine learning architecture to process the execution, perform, and tune the trajectory. However, both works does not consider the usage of multiple UAV working together to conquer a common goal, e.g., patrolling a set of POI.

Regarding swarm of UAVs, [Ryan McCune and Madey 2013] presents a review and a newer approach to the Clean Sweep problem [Wagner et al. 2008] by using multiple UAVs to perform target searches on an area. The UAV swarm runs three distinct search algorithms in order to converge on a target search. The work presents evaluations regarding the three algorithms performance and compares them. Unfortunately, it does not explore the communication model or how it impacts on the swarm performance.

In the work [Kumar et al. 2013] presents a very agile[2] swarm of UAVs, working together in several tasks such as dynamic formation, block assembling, music playing and obstacle avoidance. Its communication model relies on 802.4.15 star network with a central processing node responsible to process all movements and all UAV acts as remote controlled aircrafts. Despite our work and other above related, this work runs only in indoor areas under a camera vision system and cannot supposed to work outdoor.

6. Conclusion and Future Works

This work proposed and detailed a framework for coordination swarm of UAV relying on smartphones and mobile networks. This framework is capable to establish swarm of UAVs controlled by one user with a detailed distributed algorithm which was evaluated. This evaluation tested its accuracy against several mobile networks usual latencies and showed that is possible to maintain such swarm of UAV until latencies around $200ms$ which is acceptable for 3G and 4G technologies. Further, based on results reached, we aim to prosecute with prototyping and physical evaluations with a real swarm of UAVs.

References

[Austin 2010] Austin, R. (2010). *Unmanned Aircraft Systems*. John Wiley & Sons, Ltd, Chichester, UK, first edit edition.

[2]Video available on https://www.youtube.com/watch?v=YQIMGV5vtd4

[Barton and Kingston 2013] Barton, K. and Kingston, D. (2013). Systematic surveillance for UAVs: A feedforward iterative learning control approach. *2013 American Control Conference*, pages 5917–5922.

[Darrah et al. 2015] Darrah, M., Wilhelm, J., Munasinghe, T., Duling, K., Yokum, S., Sorton, E., Rojas, J., and Wathen, M. (2015). A Flexible Genetic Algorithm System for Multi-UAV Surveillance: Algorithm and Flight Testing. *Unmanned Systems*, 03(01):49–62.

[David et al. 2013a] David, L., Endler, M., and Roriz, M. (2013a). MR-UDP: Yet another Reliable User Datagram Protocol, now for Mobile Nodes. Technical report, Pontifícia Universidade Católica do Rio de Janeiro, MCC-06/13, ISSN 0103-9741.

[David et al. 2013b] David, L., Vasconcelos, R., Alves, L., André, R., and Endler, M. (2013b). A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes. *Journal of Internet Services and Applications*, 4(1):16.

[EsperTech 2015] EsperTech (2015). EsperTech - Complex Event Processing.

[Kumar et al. 2013] Kumar, V. U. o. P., Bhattacharya, S., and Ghrist, R. (2013). Multi-robot Coverage and Exploration on Riemannian Manifolds with Boundary. *International Journal of Robotics Research*.

[Murray 2007] Murray, R. M. (2007). Recent Research in Cooperative Control of Multivehicle Systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571.

[Olivieri and Endler 2014] Olivieri, B. and Endler, M. (2014). An ubiquitous based approach for Movement Coordination of Swarms of Unmanned Aerial Vehicles using mobile networks. In *V Simpósio de Computação Ubíqua e Pervasiva, 2014, Brasília. (SBC)*, number JULY.

[Olivieri and Endler 2015] Olivieri, B. and Endler, M. (2015). Coordinating Movement within Swarms of UAVs through Mobile Networks. In *6th IEEE Workshop on Pervasive Collaboration and Social Networking, (PerCol), IEEE PerCom Workshops, At Saint Louis*, number MARCH, pages 154–159.

[Pal et al. 2005] Pal, A., Kshemkalyani, A. D., Kumar, R., and Gupta, A., editors (2005). *Distributed Computing – IWDC 2005*, volume 3741 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Pascarella et al. 2013] Pascarella, D., Venticinque, S., and Aversa, R. (2013). Agent-based design for UAV mission planning. *Proceedings - 2013 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2013*, pages 76–83.

[Prencipe 2005] Prencipe, G. (2005). The Effect of Synchronicity on the Behavior of Autonomous Mobile Robots. *Theory of Computing Systems*, 38(5):539–558.

[Ryan McCune and Madey 2013] Ryan McCune, R. and Madey, G. R. (2013). Swarm control of UAVs for cooperative hunting with DDDAS. *Procedia Computer Science*, 18:2537–2544.

[Wagner et al. 2008] Wagner, I. A., Altshuler, Y., Yanovski, V., and Bruckstein, A. M. (2008). Cooperative Cleaners: A Study in Ant Robotics. *The International Journal of Robotics Research*, 27(1):127–151.

[Şahin and Spears 2005] Şahin, E. and Spears, W. M., editors (2005). *Swarm Robotics*, volume 3342 of *LNCS*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Solução Ótima para Alocação de Recursos de Rádio com Restrições de QoS em Sistemas Multicelulares com Múltiplos Serviços

Lászlon R. Costa[1,2], F. Rafael M. Lima[1,2], Yuri C. B. Silva[1], F. Rodrigo P. Cavalcanti[1]

[1]Grupo de Pesquisa em Telecomunicações Sem Fio (GTEL), Universidade Federal do Ceará, Fortaleza, Brasil

[2]Programa de Pós-Graduação em Engenharia Elétrica e de Computação (PPGEEC), Universidade Federal do Ceará, Sobral, Brasil

{laszlon,rafaelm,yuri,rodrigo}@gtel.ufc.br

Resumo. *Este trabalho propõe uma solução ótima para distribuição de RB (do inglês, Resource Block) considerando os efeitos da interferência, com o objetivo de maximizar a soma da taxa de dados total de um sistema multi celular com múltiplos serviços e restrições que assegurem um percentual mínimo de usuários satisfeitos para cada serviço. De forma a atingir uma experiência uniforme na área de cobertura, consideraremos critérios de satisfação de usuários localizados no centro e borda da célula. Resultados obtidos através de simulações computacionais mostram a importância de aplicar estratégias de gerenciamento de interferência em sistemas modernos e demonstram as vantagens da solução do problema formulado quando comparado às soluções encontradas na literatura.*

Abstract. *This work proposes an optimal solution to the Resource Block (RB) distribution in order to maximize the sum rate in a multicell and multiservice system with constraints on the percentage of users that should be satisfied per service. In order to achieve a more uniform experience in the coverage we propose takes into account the satisfaction of users located at the center and edge of cells. Results of computer simulations show the importance of interference management in modern systems and the advantages of proposed solution when compared with others solutions found in the literature.*

1. Introdução

Com a rápida popularização e adesão ao uso de dispositivos móveis inteligentes como *smartphones*, *tablets* e *laptops* portáteis, têm crescido também a demanda por altas taxas de dados e a necessidade do uso de diversos serviços em redes móveis. Para atender esta crescente demanda, as operadoras devem garantir que suas redes suportem o aumento do tráfego de dados, o que tem se mostrado cada vez mais desafiador. O desenvolvimento de novas estratégias para suprir o aumento da demanda de tráfego de dados estão em consonância com o desenvolvimento da Quinta Geração (5G) de redes móveis.

O aumento da capacidade de uma rede celular pode ser alcançado basicamente por três métodos: alocação de novas bandas de frequência, otimizar o uso da camada física do sistema ou novos paradigmas para a arquitetura da rede celular. O espectro de frequência que atualmente é utilizado em sistemas móveis é escasso, além do fato de que o aumento dessa banda pode ser custoso para as operadoras que devem enfrentar entraves burocráticos para concessão de mais bandas de frequência. Melhorias na camada

física, como novos esquemas de codificação e modulação, transmissão e recepção ou mesmo técnicas de múltiplo acesso estão chegado cada vez mais perto do limite teórico da eficiência espectral. Sendo assim, melhorias na arquitetura da rede se mostram como um dos principais meios para melhorar o desempenho das tecnologias de comunicações para a próxima geração [Bhushan et al. 2014].

Um método para aumentar a capacidade e atender grandes volumes de tráfego tal qual ocorre em regiões muito populosas, consiste em diminuir a distância entre as Estações Rádio Base (ERBs). Considerando uma densidade de usuários fixa, essa técnica garantirá que as ERBs servirão menos terminais e necessitarão de menos recursos de rádio. Os recursos mais relevantes em um sistema de comunicação são a potência de transmissão, banda de frequência e *time slot*, que são gerenciados de forma a atingir um melhor objetivo como maximização de taxa de transmissão, redução do atraso ou mesmo diminuição dos erros de bits. Na densificação teremos mais recursos de rádio disponíveis por área, ou seja, é possível maior reuso de recursos aumentando a capacidade de usuários. A densificação é uma das principais linhas de pesquisa para o 5G [Bhushan et al. 2014].

Entretanto, a maior densidade de canais por área obtida através da densificação da rede apresenta desafios relacionados ao aumento de interferência e carga de sinalização. Uma rede com distâncias entre ERBs reduzidas também implica em efeitos de perda de percurso menores, tornando comum casos que o sinal de interferência tenha a mesma ordem de grandeza dos sinais de interesse. Usuários localizados nas bordas das células tendem a receber maior interferência e potência de interesse mais fraca, requisitando assim, uma atenção especial. Sendo assim, a distribuição de recursos deve ser cuidadosamente planejada e executada de forma a combater os efeitos colaterais de densificação de ERBs. Uma forma de garantir uma boa gestão global de recursos na rede é através de alguma entidade central que, com total conhecimento dos requisitos e recursos do sistema, gerencie a rede baseado nas necessidades dos usuários e recursos disponíveis. As considerações deste trabalho podem ser implementadas na arquitetura de rede C-RAN (do inglês, *Cloud Radio-Acess Network*) que utiliza processamento centralizado para múltiplas ERBs [Mohsen and Hassan 2015].

2. Estado da Arte e Contribuições

Problemas de RRA (do inglês, *Radio Resource Allocation*) são formulados como problemas de otimização que são compostos de uma função objetivo e restrições [Sadr et al. 2009, Bohge 2010].

Há na literatura um grande volume de trabalhos para alocação de recursos com diversos objetivos, porém sua grande maioria aplicada em cenário com uma única célula desprezando, portanto, os efeitos da interferência intercelular. Alocação de RB (do inglês, *Resource Block*) em sistemas OFDMA (do inglês, *Orthogonal Frequency Division Multiple Access*) em um cenário com multicelulas foi estudado no trabalho [Rahman and Yanikomeroglu 2007] em que é proposto um problema de alocação de RBs com o objetivo de maximizar a soma da taxa de dados para um sistema que oferece um único serviço com apenas a taxa de dados como requisito de QoS (do inglês, *Quality of Service*) em termos de taxa de dados. Foi utilizado um conceito de grupos interferentes, que é composto por células que utilizam os mesmos recursos e possuem antenas com padrão de irradiação dominante na direção da área de cobertura. No cenário abordado não há um limite de tolerância mínimo para o número de usuários que devem ser satisfeitos.

Uma análise entre taxa de dados e QoS em termos de atrasos de pacotes para um sistema com multicelulas é proposto em [Akbari and Vakili 2012]. O problema formulado é não convexo e combinatorial e nenhuma análise em relação a solução ótima para o problema é apresentada. O trabalho [Rahman and Yanikomeroglu 2008] propõe uma solução subótima semi distribuída em uma arquitetura hexagonal setorizada para maximização de taxa de dados em que é levada em conta a interferência de apenas dois anéis interferentes. A proposta utiliza o algoritmo Húngaro [Kuhn 2010] em cada ERB e após este passo uma resolução de conflitos é feita por uma entidade central.

Abordagens que também consideram a associação entre usuários e ERBs em conjunto com a alocação de recursos para maximização da mínima taxa dos usuários no sistema são analisadas em [Sadr and Adve 2014] e [Gotsis and Alexiou 2013]. Especificamente em [Sadr and Adve 2014], os autores propõem a utilização de quatro passos para este objetivo, sendo eles: (1) a associação de usuários com ERBs, (2) estimação da carga de acordo com a taxa exigida pelos usuários, (3) distribuição de recursos entre as ERBs através de coloração de grafos e (4) alocação dos recursos para os usuários com informações locais. Múltiplos serviços e um número mínimo de usuários a serem satisfeitos não são considerados, além de que o escalonamento de recursos realizado pelas ERBs, no quarto passo, não consideram a interferência do sistema. A eficiência da solução baseia-se na distribuição dos recursos entre as ERBs no terceiro passo, evitando que células vizinhas usem o mesmo recurso. Uma abordagem diferente é estudada em [Gotsis and Alexiou 2013], que propôs uma formulação ótima para um problema de maximização da SINR (do inglês, *Signal to Interference plus Noise Ratio*) mínima do sistema através da associação de usuários, alocação de RBs e alocação de potência. O algoritmo tenta solucionar de forma ótima um ILP (do inglês, *Integer Linear Program*) dado uma SINR mínima alvo. O maior valor possível da SINR alvo é encontrada através do método da bisseção, sendo necessário resolver o problema ILP para cada iteração. Toda a formulação é baseada em SINR alvo mínima e não considera restrições de taxa. O método torna possível que o sistema encontre a melhor SINR mínima, porém não há garantias que os recursos alocados garantam taxas de dados necessária para uso de serviços. Ambos os trabalhos [Sadr and Adve 2014] e [Gotsis and Alexiou 2013] concluem que a associação ótima entre usuários e ERBs é próxima de uma associação baseada apenas em qualidade de canal.

A redução da interferência em cenário com multicelulas pode ser alcançada através da minimização da potência de transmissão das células do sistema. O trabalho [Lopez-Perez et al. 2014] propõe um problema de otimização ILP com objetivo de minimizar a potência total transmitida no sistema com restrições de QoS dos usuários em termos de taxa de dados. O problema formulado é de alta complexidade e os autores propõem utilizar uma abordagem distribuída, em que o problema é resolvido localmente e de forma independente por cada ERB. Este método espera que com a redução contínua de potência irradiada no sistema, a interferência seja mitigada levando a uma convergência em que a rede se auto organize. O método não traz uma solução ótima e o problema formulado só apresentará a solução ótima se houver apenas uma célula no sistema.

De acordo com nossa pesquisa bibliográfica, não encontramos na literatura uma proposta de problema de otimização que considere a heterogeneidade de requisitos de QoS de diferentes serviços de tráfego e que permita a satisfação de um número mínimo de usuários por serviço. Uma solução com essas características para um cenário com uma

única célula foi proposto em [Lima et al. 2012] em que é formulado um novo problema de RRA para maximização da taxa total do sistema restrito a requisitos de satisfação mínimas por cada serviço oferecido no sistema.

O objetivo deste trabalho é propor uma generalização do problema formulado em [Lima et al. 2012] para um ambiente com multicelulas limitado por interferência. O novo problema de otimização também explorará os requisitos de QoS em termos de taxa de dados criando justiça para usuários de borda em cenários com células hexagonais. Além da nova formulação este trabalho propõe uma solução ótima. Simulações computacional foram realizadas, evidenciando a vantagem em considerar interferência ao realizar alocação de recursos.

Este artigo segue a seguinte organização. Na seção 3 apresentamos o cenário e as considerações adotadas neste trabalho. A formulação do problema e a caracterização da solução ótima é apresentada na seção 4. As simulações computacionais e análise dos resultados estão presentes na seção 5. Por fim, as conclusões deste trabalho estão apresentadas na seção 6.

3. Modelagem do Sistema

Neste trabalho consideramos um sistema TDD (do inglês, *Time Division Duplexing*) composto por M células, utilizando antenas omnidirecionais, que servem um total de J usuários. Cada usuário pode utilizar um dos S serviços de tráfego oferecidos pela operadora. Como exemplo de serviços temos VoIP (do inglês, *Voice over IP*) e *web browsing*. O trabalho é baseado no enlace direto e o acesso dos usuários é realizado por meio de uma combinação de OFDMA e TDMA (do inglês, *Time Division Multiple Access*). O recurso mínimo a ser alocado durante um TTI (do inglês, *Transmission Time Interval*) é o RB que é composto por uma combinação de subportadoras e símbolos OFDMA adjacentes. A qualidade de um RB é medida através de sua SINR, que mede a relação entre a potência do sinal recebido pela soma da potência do sinal interferente adicionado ao ruído térmico do sistema. Para uma modelagem coerente será considerado que todas as subportadoras de um mesmo RB tenham ganhos de canais correlacionados.

Seja $\mathcal{J}_m^{\text{ERB}}$ o conjunto de usuários que é servido por uma ERB m e $\mathcal{J}_n^{\text{RB}}$ o conjunto de usuários que utilizam o RB n, podemos modelar a SINR observada pelo usuário j servido pela ERB m no RB n como:

$$\gamma_{j,n} = \frac{p^{\text{RB}}\, h_{j,m}\, |\alpha_{j,m,n}|^2}{\sigma^2 + \sum\limits_{m' \neq m}^{M} p^{\text{RB}}\, h_{j,m'}\, |\alpha_{j,m',n}|^2}, \tag{1}$$

em que $h_{j,m}$ é a perda de percurso com o desvanecimento de longo prazo entre um usuário j e uma ERB m enquanto que o desvanecimento rápido desse *link* em um dado RB n é representado por $\alpha_{j,m,n}$. A potência do ruído e de transmissão em um RB são respectivamente σ^2 e p^{RB}.

A taxa de dados de transmissão em uma comunicação sem fio depende do estado do canal e potências de transmissão, ruído e sinais interferentes. Considere que o mapeamento entre a qualidade de canal e taxa de dados obtida na camada física é dada por uma função de *link* adaptativo $f(.)$. Em sistemas reais a adaptação de enlace leva em conta o MCS (do inglês, *Modulation and Coding Scheme*) que mapeia a taxa de dados de acordo com a SINR e a taxa de erro de bloco, ou BLER (do inglês, *Block Error Rate*). Quanto maior o nível de MCS maior a capacidade de transmitir bits, porém também serão maiores

Tab. 1. Mapeamento geral entre SINR e taxa de dados transmitida por RB.

Região de SINR	Taxa de dados transmitida por RB
$\gamma_{j,n} < \gamma^1$	0
$\gamma^1 \leq \gamma_{j,n} < \gamma^2$	r^1
$\gamma^2 \leq \gamma_{j,n} < \gamma^3$	r^2
$\vdots$	$\vdots$
$\gamma^{V-1} \leq \gamma_{j,n} < \gamma^V$	r^{V-1}
$\gamma_{j,n} \geq \gamma^V$	r^V

as probabilidades de erro de bloco em função da SINR. Sendo assim quanto maior o nível de MCS mais alta será a SINR necessária para garantir uma taxa de erro mínima aceitável que possibilite uma transmissão satisfatória. Desta forma, a taxa de dados total de um usuário j, usando um recurso n em um nível de MCS v é dada por:

$$r_{j,n,v} = BR_v \cdot (1 - \text{BLER}(v, \gamma_{j,n})) \,, \tag{2}$$

em que BR_v é a taxa de bits bruta em um nível de MCS v e BLER é a função que mapeia a probabilidade de erro de blocos de bit levando em conta o nível de MCS e a SINR. Sem perder a generalidade consideraremos que para um usuário usar um nível de MCS$_v$ será necessário atingir um nível de SINR em que a BLER seja tolerável. A taxa de dados de um usuário j em um RB n será mapeada de forma discreta por uma função adaptativa escada $f(\gamma_{j,n})$, como ilustrado na Tabela 1 em que γ^v e r^v são respectivamente a SINR mínima tolerável para atingir uma MCS$_v$ e a taxa de dados referente a MCS$_v$. Para este trabalho será considerado o esquema de MCS do LTE (do inglês, *Long Term Evolution*) [3GPP 2009], composta por 15 níveis.

O sistema considerado neste trabalho oferece múltiplos serviços, tornando possível o usuário utilizar a rede para diferentes fins, como por exemplo: serviços de dados, voz, mensagens de texto e serviços de emergências. Cada serviço possui diferentes necessidades, e como consequência há diferentes critérios de qualidade. Os critérios de QoS mais comuns em redes móveis são taxa de dados, tempo de atraso de pacotes, confiabilidade de dados, etc. Neste contexto, um sistema que oferece diversos serviços deve alocar seus recursos de acordo com as necessidades dos serviços utilizados pelos usuários, evitando desperdícios de recursos e procurando atender critérios da satisfação. Neste trabalho consideraremos apenas requisitos de taxa para múltiplos serviços, em que cada serviço oferecido será associado a uma taxa de dados.

Buscando manter justiça no acesso a rede entre usuários de centro e de borda das células hexagonais, ou seja, garantir para ambos os tipos de usuários os mesmos requisitos de taxa de dados, iremos considerar uma duplicação de cada serviço oferecido. Embora um usuário de borda e de centro possam utilizar o mesmo tipo de serviço faremos uma distinção entre estes. Por exemplo: considere uma operadora que oferece dois serviços s_1 e s_2 com exigências de taxa t^{s_1} e t^{s_2} respectivamente. Na prática modelaremos para o problema quatro serviços virtuais s_1, s_2, s_3 e s_4 com requisitos de taxa t^{s_1}, t^{s_2}, t^{s_1} e t^{s_2}, respectivamente. A diferença nessa formulação é que os usuários dos serviços s_3 e s_4 serão apenas usuários de borda enquanto s_1 e s_2 são exclusivos para usuários do centro. Diversos critérios podem ser utilizados para classificar usuários de centro e borda da célula. Como exemplo podemos citar geolocalização ou potência média recebida. Por simplicidade, neste trabalho iremos considerar que a localização geográfica do terminal móvel é

conhecida. Será mostrado nas próximas seções que esta criação de serviços virtuais resultará em ganhos de justiça em termos de número de usuários satisfeitos na borda da célula. A Figura 1 faz uma representação gráfica da rede mostrando o limitante entre centro e borda da célula. Para a aplicação proposta de solução ótima é necessário que a entidade

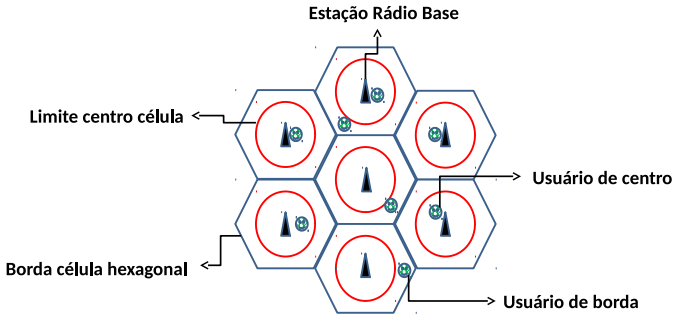


Figure 1. Representação gráfica da rede com divisão entre centro e borda das células.

central que determinará a alocação de recursos tenha conhecimento, além dos ganhos de canais entre usuários e ERB servidora, o estado de canal entre terminais e ERBs interferentes. Para isto, consideraremos que o sistema é capaz de estimar os canais interferentes. Essa estimativa é possível utilizando SRS (do inglês, *Sounding Reference Signal*), que foi definido no padrão LTE-A (do inglês, *Long Term Evolution - Advanced*) [3GPP 2012]. Este método foi projetado para se ter uma estimativa de ganhos de canal em qualquer recurso de frequência, mesmo nos que não estejam sendo utilizados para transmissão. Periodicamente os terminais enviam sinais de referência baseadas na sequência de Zadoff-Chu que quando deslocadas no tempo têm a propriedade da ortogonalidade entre cada cópia deslocada. Para mais detalhes convidamos a leitura de [Heimiller 1961] e [Chu 1972].

Considerando área densa de baixa mobilidade e sujeito a uma entidade central administradora da rede, é possível alocar diferentes *shifts* de uma mesma sequência de Zadoff-Chu por vez, dentro de um *cluster* cooperativo de Estação Rádio Bases (ERBs), respeitando o tempo de coerência dos canais no ambiente. Assim, quando um usuário na transmissão do canal reverso enviar um sinal de referência ortogonal para a ERB servidora, todas as ERBs interferentes poderão isolar este sinal e assim estimar o ganho da interferência que causarão nestes usuários. Para os resultados apresentados nas próximas seções, estaremos interessados na solução ótima e consideramos o perfeito conhecimento dos estados de canais. Análise da robustez dos algoritmos envolvidos na presença de incerteza em relação ao conhecimento da qualidade de canal são tópicos a serem explorados em trabalhos futuros.

4. Formulação do Problema e Solução Ótima

Consideraremos um sistema com potência de transmissão fixa para cada RB. Esta é uma consideração coerente, pois em geral na literatura é preferido um assinalamento

de recursos com potência fixa que soluções com alocações conjuntas de potência devido a alta complexidade dos problemas, tratamento matemático e dificuldades de implementação [Lopez-Perez et al. 2014]. Análises prévias feitas em [Song and Li 2005] e [Jang and Lee 2003] mostram que a melhora causada pela alocação de canais com potência diferentes é negligenciável quando aplicado em cenários com grande número de usuários e diversos requisitos de QoS. Baseado nestes resultados procuraremos obter ganhos na taxa de transmissão através apenas da gestão na distribuição de RBs, levando em conta a interferência e o ganho de canal.

A alocação de RBs para os usuários servidos por uma dada ERB é realizado de forma exclusiva, ou seja, em uma célula cada RB poderá ser alocado a no máximo um usuário, eliminando assim interferência intracelular. Sendo assim apenas a interferência intercelular poderá estar presente. Para contabilizar toda a interferência no sistema usaremos o conceito de grupos interferentes. Seja $\mathcal{G}$ o conjunto de todos os grupos interferentes que é formado por todas as possíveis combinações entre usuários, incluindo grupos com um único usuário. Por exemplo, considere que um sistema tenha três ERBs servindo quatro usuários dispostos na forma de: $\mathcal{J}_1^{\text{ERB}} = \{1\}$, $\mathcal{J}_2^{\text{ERB}} = \{2, 3\}$ e $\mathcal{J}_3^{\text{ERB}} = \{4\}$. Note que não pode haver alocações de um mesmo RB para os usuários 2 e 3, pois ambos são servidos pela mesma ERB. Sendo assim devemos excluir grupos que contenham estes dois usuários. As combinações válidas de interferentes serão $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$, $\{2, 4\}$, $\{3, 4\}$, $\{1, 2, 4\}$ e $\{1, 3, 4\}$. O número mínimo de grupos interferentes válidos para um sistema é equivalente a quantidade de usuários, se considerarmos que há apenas uma ERB para todos os usuários. Portanto, quanto mais densa a rede menos grupos interferentes. O caso de uma única ERB corresponde ao cenário adotado em [Lima et al. 2012].

Para formulação do problema de otimização devemos definir variáveis que representem associações entre as entidades no sistema. A associação entre um usuário j e uma ERB m será representada pela variável binária $z_{j,m}$ que assumirá valor 1 caso j seja servido pela ERB m e 0 caso contrário. Neste trabalho a associação entre usuários e ERB será realizada através da qualidade média do canal, ou seja, o usuário será servido pela ERB que apresente o melhor estado de canal médio, tornando $z_{j,m}$ uma constante que pode ser pré-calculada antes resolver o problema de otimização. De forma semelhante a participação de um usuário j em um grupo interferente g será representado pela variável binária $o_{g,j}$ e será calculada antes da aplicação da solução do problema. Assim $o_{g,j}$ assumirá valor 1 quando o usuário j estiver presente no grupo g e 0 caso contrário. A taxa de dados em um único bloco de recurso n de um usuário j que participa de um grupo g será representada por $r_{g,j,n}$. Note que como sabemos quais os usuários que fazem parte de um grupo interferente e conhecemos também os ganhos de canais de interesse e interferentes podemos utilizar a Equação (1) para determinar a SINR dos participantes do grupo e mapear suas taxas de dados. Essa arquitetura só é possível se considerarmos que a solução é realizada por uma unidade central que recebe os dados de todas as ERBs e define a alocação em todo o sistema. Definimos também a variável $c_{j,s}$ que assumirá valor 1 caso o usuário j esteja utilizando o serviço s e 0 caso contrário. Neste trabalho consideraremos que cada usuário usará apenas um serviço por vez. A escolha de qual recurso será utilizado por um grupo é a variável binária de otimização $x_{g,n}$ que assume valor 1 para associação entre o grupo g e RB n, e 0 caso contrário. O problema é formulado como:

$$\max_{x_{g,n}} \sum_{j} \sum_{n} \sum_{g} x_{g,n}\, r_{g,j,n}\, o_{g,j} \tag{3a}$$

$$\text{sujeito a } \sum_{g} x_{g,n} = 1 \quad \forall\, n, \tag{3b}$$

$$\sum_{j} u\left(\sum_{g} \sum_{n} x_{g,n}\, o_{g,j}\, r_{g,j,n}\, c_{j,s}\, z_{j,m},\, r^{j}_{\text{obj}} \right) \geq k_s \quad \forall s \text{ e } \forall m, \tag{3c}$$

$$\sum_{j} \sum_{g} x_{g,n}\, o_{g,j}\, z_{j,m} \leq 1 \quad \forall\, m \text{ e } \forall\, n, \tag{3d}$$

$$x_{g,n} \in \{0, 1\} \quad \forall\, n \text{ e } \forall\, g. \tag{3e}$$

A função objetivo (3a) é a soma da taxa de dados de todos os usuários e todos os RBs no sistema. A restrição (3b) garante que um recurso apenas será alocado para um único grupo de interferência, desse modo essa restrição garante que a escolha da alocação sempre contabilizará todos os efeitos de interferência possíveis. A segunda restrição (3c) é o requisito de QoS em que $u(b, d)$ é a função degrau que assumirá valor 1 caso $b \geq d$ e 0 caso contrário. Nesta restrição r^{j}_{obj} representa a taxa alvo para o usuário j atingir a satisfação. Esta taxa depende do serviço utilizado em que cada serviço tem sua própria taxa mínima necessária. O número mínimo de usuários a serem satisfeitos que usam o serviço s é dado por k_s. A última restrição (3d) garante a exclusividade no uso dos RBs dentro da área de cobertura de cada ERB. O problema formulado anteriormente tem a função objetivo e todas as restrições lineares, com exceção da restrição (3c) pois a função degrau não é linear. Por conta da não linearidade do problema, a solução ótima só poderá ser encontrada por busca exaustiva, que possui alta complexidade computacional. Porém, podemos linearizar o problema adicionando uma nova variável ρ_j que assume o valor 1 caso o usuário j esteja satisfeito em relação a seu QoS, ou seja, $\sum_g \sum_n x_{g,n}\, o_{g,j}\, r_{g,j,n} \geq r^{j}_{\text{obj}}$ e 0 caso contrário. A reformulação do problema será:

$$\max_{x_{g,n}, \rho_j} \sum_{j} \sum_{n} \sum_{g} x_{g,n}\, r_{g,j,n}\, o_{g,j} \tag{4a}$$

$$\text{sujeito a } \sum_{g} x_{g,n} = 1 \quad \forall\, n, \tag{4b}$$

$$\sum_{g} \sum_{n} x_{g,n}\, o_{g,j}\, r_{g,j,n} \geq \rho_j\, r^{j}_{\text{obj}} \quad \forall\, j, \tag{4c}$$

$$\sum_{j} \rho_j\, c_{j,s}\, z_{j,m} \geq k_s, \quad \forall\, s \text{ e } \forall\, m, \tag{4d}$$

$$\sum_{j} \sum_{g} x_{g,n}\, o_{g,j}\, z_{j,m} \leq 1 \quad \forall\, m \text{ e } \forall\, n, \tag{4e}$$

$$x_{g,n} \in \{0, 1\} \quad \forall\, n \text{ e } \forall\, g, \tag{4f}$$

$$\rho_j \in \{0, 1\} \quad \forall\, j. \tag{4g}$$

Note que o novo problema segue a mesma estrutura formulada anteriormente, porém, a restrição (3c) foi substituída pelas restrições (4c) e (4d), em que respectivamente, é garantido a taxa mínima para satisfação dos usuários selecionados e o número

mínimo de terminais satisfeitos por serviço. O problema é combinatorial da classe ILP e a solução ótima agora pode ser encontrada utilizando métodos como o algoritmo BB (do inglês, *Branch-and-Bound*) [Nemhauser and Wolsey 1988]. A complexidade deste algoritmo cresce exponencialmente com número de restrições e variáveis. Caso um problema tenha l variáveis e p restrições o número de subproblemas lineares a serem resolvidos será de $(\sqrt{2})^l$. Cada subprolema necessitará de $2(p+l)$ iterações em que serão executados $(pl-p)$ multiplicações, $(pl-p)$ somas e $(l-p)$ comparações [Zhang and Letaief 2004, Sierksma 2001]. Na formulação proposta temos $NG+J$ variáveis e $N+J+M(S+N)$ restrições, em que G e N são o número de grupos e RBs no sistema. Sendo assim o número de operações para solução no pior caso é dado por

$$\sqrt{2}^{NG+J}2(NG{+}2J{+}N{+}M(S{+}N))(2(NG{+}J)(N{+}J{+}M(S{+}N))-3(N{+}J{+}M(S{+}N))+NG{+}J), \tag{5}$$

considerando o termo de maior ordem a complexidade computacional, no pior caso, será $\mathcal{O}(2^{NG})$.

Para utilização de *softwares* para solução de problemas de otimização, como é o caso do CPLEX [IBM 2009], é necessário transcrever o problema para uma forma mais compacta e baseado em operações matriciais. Sendo assim, demonstraremos abaixo uma construção de matrizes para o problema (4) em forma compacta com apenas uma restrição matricial de igualdade e uma de desigualdade.

Sendo $(\cdot)^T$ a transposição de uma matriz e agrupando alguns valores do problema em vetores poderemos agrupar as variáveis de otimização na forma de $\mathbf{x}_g = [x_{g,1}, \cdots, x_{g,N}]$, $\mathbf{x} = \left[\mathbf{x}_1^T, \cdots, \mathbf{x}_g^T\right]^T$. Usaremos uma nova variável de otimização definida como $\mathbf{y} = \left[\mathbf{x}^T|\boldsymbol{\rho}^T\right]$. Para isolar as variáveis desejadas precisamos aplicar filtros que torna possível as operações: $\mathbf{x} = \mathbf{A}_1\,\mathbf{y}$ e $\boldsymbol{\rho} = \mathbf{A}_2\,\mathbf{y}$. Os filtros que mantêm essa propriedade são $\mathbf{A}_1 = [\mathbf{I}_{G\,N}|\mathbf{0}_{G\,N\times J}]$ e $\mathbf{A}_2 = [\mathbf{0}_{J\times G\,N}|\mathbf{I}_J]$ em que $\mathbf{I}_J$ é uma matriz identidade de dimensão $J\times J$ e $\mathbf{0}_{n_1\times n_2}$ uma matriz de zeros de dimensão $n_1\times n_2$. Considere também os vetores $\mathbf{o}_g = [o_{g,1}, o_{g,2}, \cdots, o_{g,J}]^T$, $\mathbf{r}_{g,n} = [r_{g,1,n}, \cdots, r_{g,J,n}]^T$ e $\mathbf{a} = \left[\mathbf{o}_1^T\,\mathbf{r}_{1,1}, \cdots, \mathbf{o}_1^T\,\mathbf{r}_{1,N}|\cdots|\mathbf{o}_G^T\,\mathbf{r}_{G,1}, \cdots, \mathbf{o}_G^T\,\mathbf{r}_{G,N}\right]$. Sendo assim a nova função objetivo será $\mathbf{a}\,\mathbf{A}_1\,\mathbf{y}$.

Para as restrições também precisaremos definir novos vetores. A associação usuário-ERB é representada pelo vetor binário $\mathbf{z}'_m = [z_{m,1}, \cdots, z_{m,J}]$. Também será necessário definir um vetor binário com a associação entre usuário-ERB-serviço que será composta pelo vetor $\mathbf{z}_{m,s} = [z_{m,1,s}, \cdots, z_{m,J,s}]^T$ em que $z_{m,j,s}$ assume valor um caso o usuário j esteja sendo servido pela ERB m e usando o serviço s. O número de usuários mínimos a serem satisfeitos em cada serviço e em cada ERB será dado pelo vetor $\mathbf{k} = \left[\underbrace{k_1, \cdots, k_S|\cdots|k_1, \cdots, k_S}_{M \text{ vezes}}\right]^T$.

Como não poderemos alocar recursos para grupos com mais de um usuário da mesma ERB precisaremos definir uma função auxiliar que determina se um grupo possui mais de um usuário servido pela mesma ERB. Matematicamente essa função será:

$$w_g := \begin{cases} 0, & \text{se } \sum_m u(\mathbf{o}_g^T\,\mathbf{z}'_m, 2) > 0, \\ 1, & \text{caso contrário.} \end{cases} \tag{6}$$

Com isso o vetor que usaremos para determinar a validade de um grupo interferente na nossa formulação será $\mathbf{w} = \left[\underbrace{w_1, \cdots, w_1}_{N\text{vezes}} | \cdots | \underbrace{w_G, \cdots, w_G}_{N\text{vezes}}\right]^T$. Com todos os vetores formulados acima devemos definir a partir delas matrizes que serão: $\mathbf{B} = \left[\underbrace{\mathbf{I}_N, \cdots, \mathbf{I}_N}_{G\text{ vezes}}\right]$, $\mathbf{C} = \begin{bmatrix} o_{1,1}r_{1,1,1} \cdots o_{1,1}r_{1,1,N} | \cdots | o_{G,1}\, r_{G,1,1} \cdots o_{G,1}\, r_{G,1,N} \\ \vdots \quad \ddots \quad \vdots \\ o_{1,J}r_{1,J,1} \cdots o_{1,J}r_{1,J,N} | \cdots | o_{G,J}\, r_{G,J,1} \cdots o_{G,J}\, r_{G,J,N} \end{bmatrix}$, $\mathbf{Z} = \begin{bmatrix} \mathbf{z}_{1,1} & \cdots & \mathbf{z}_{1,S} | \cdots | \mathbf{z}_{M,1} & \cdots & \mathbf{z}_{M,S} \end{bmatrix}^T$, e $\mathbf{E} = \text{diag}(r^1_{\text{obj}}, \cdots, r^J_{\text{obj}})$, em que $\text{diag}(\cdot)$ é a função que representa uma matriz diagonal em que a diagonal principal são os valores de entrada e $\mathbf{1}_{J_s}$ é um vetor coluna composto de 1's com o tamanho J_s que é o número de usuários utilizando o serviço s.

Baseado nas definições matriciais acima podemos redefinir a restrição (4b) como $\mathbf{B}\,\mathbf{A}_1\,\mathbf{y} = \mathbf{1}_N$, em quem $\mathbf{1}_N$ é um vetor coluna composto por 1's de dimensão N. A restrição (4c) será reformulada como $\mathbf{C}\,\mathbf{A}_1\,\mathbf{y} \geq \mathbf{E}\,\mathbf{A}_2\,\mathbf{y} \Rightarrow (\mathbf{C}\,\mathbf{A}_1\,\mathbf{y} - \mathbf{E}\,\mathbf{A}_2\,\mathbf{y}) \geq \mathbf{0}_J$. As restrições (4d) e (4e) serão respectivamente $\mathbf{Z}\,\mathbf{A}_2\,\mathbf{y} = \mathbf{k}$ e $\mathbf{A}_1\,\mathbf{y} \leq \mathbf{w}$.

Note que temos duas restrições de igualdade e duas de desigualdade. Para compactarmos o problemas iremos concatená-las em apenas duas restrições, uma de cada classe. Considere as concatenações $\mathbf{U} = \left[(\mathbf{B}\,\mathbf{A}_1)^T \,|\, (\mathbf{Z}\,\mathbf{A}_2)^T\right]^T$, $\mathbf{P} = \left[-\left(\mathbf{C}\,\mathbf{A}_1 - \mathbf{E}\,\mathbf{A}_2\right)^T | \mathbf{A}_1^T\right]^T$, $\mathbf{v} = \left[\mathbf{1}_N^T | \mathbf{k}^T\right]^T$ e $\mathbf{q} = \left[\mathbf{0}_J^T | \mathbf{w}^T\right]^T$, então podemos reformular todo o problema como:

$$\max_{\mathbf{y}} \quad \mathbf{a}\,\mathbf{A}_1\,\mathbf{y} \tag{7a}$$

sujeito a

$$\mathbf{U}\,\mathbf{y} = \mathbf{v}, \tag{7b}$$

$$\mathbf{P}\,\mathbf{y} \leq \mathbf{q}. \tag{7c}$$

5. Resultados

Para avaliar o desempenho das soluções ótima proposta realizamos simulações em diversos cenários utilizando o *software* matemático MATLAB. Para encontrar as soluções ótimas dos problemas ILP utilizamos o CPLEX como método de resolução. As soluções utilizadas para comparação foram:

- **Apenas SNR**: Usaremos este termo para referir-se a solução ótima que foi proposta em [Lima et al. 2012]. Neste problema a alocação de recursos é realizada por uma única célula considerando apenas a SNR (do inglês, *Signal-to-Noise Ratio*). Replicamos a formulação do problema ILP proposto e resolvemos o problema em cada célula sem considerar o conhecimento sobre a interferência.
- **Único Serviço**: Método que não distingue serviços para usuários de borda e centro da célula. Para viabilizar a comparação com nossa proposta, consideraremos que o número mínimo de usuários satisfeitos para um dado serviço de tráfego neste caso é igual a soma do número de usuários que devem ser satisfeitos para o mesmo serviço no centro e borda da célula.

Os principais parâmetros de simulação são apresentados na Tabela 2. Para validação estatística dos métodos utilizamos TTIs totalmente descorrelacionados, de forma que a cada TTI novas posições de usuários e ganhos de canais são gerados. Neste

caso o problema deve resolver a alocação em um *snapshot*, que é o instante em que todos os parâmetros são considerados fixos, sem variação temporal dos elementos do sistema.

Tab. 2. Parâmetros da Simulação Células Hexagonais

Parâmetro	Valor	Unidade
Raio da célula	200	m
Raio do centro da célula	150	m
Potência de transmissão por RB	8.2607	dBm
N. de subportadoras por RB	12	-
Desvio padrão do sombreamento	10	dB
Perda de Percurso	$30.6 + 36.7 \cdot \log_{10}(d)$	dB
Densidade espectral do ruído	$3.16 \cdot 10^{-20}$	W/Hz
Número de TTIs (para cada requisito de QoS)	1000	-
N. de serviços	2 (1 centro e 1 borda)	-
N. de terminais por serviço	3 (6 para **Único Serviço**)	-
N. de Células	3	-
N. de RBs	15	-
N. min. de usuários satisfeitos por serviço	2 em cada (4 para **Único serviço**)	-

Note que consideramos dois serviços para a simulação. Os mesmo requisitos de do inglês, *Quality of Service* (QoS) são aplicados em ambos, porém o serviço 1 agrupa usuários do centro da célula enquanto o serviço 2 agrupa os usuários de borda. O cenário simulado possui apenas um serviço real, porém múltiplos serviços podem ser considerados na formulação proposta na seção 4. O mesmo cenário também será aplicado para o **Único Serviço**, com a distinção que a restrição de usuários satisfeito será a soma do número mínimo de usuários de centro e de borda aplicados para a solução proposta.

Uma métrica avaliada foi a capacidade do método de resolver o problema no cenário adotado. Para isto analisaremos o *outage* ocorrido em cada TTI. Neste trabalho consideraremos que houve *outage* em um TTI caso o método não atinja o mínimo de satisfação de QoS para usuários de centro e de borda no problema proposto. Para medir probabilidade do método encontrar uma solução que respeite todas as restrições analisaremos a taxa de *outage*, que é a porcentagem de *outage* verificado nos TTIs simulados. A Figura 2 mostra o gráfico da taxa requisitada versus a taxa de *outage*.

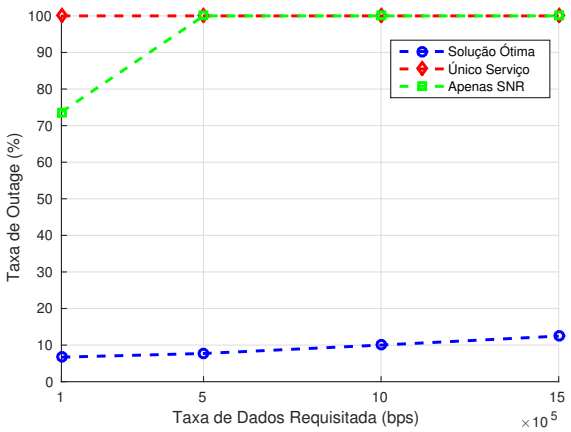


Figure 2. Taxa de *outage* versus taxa de dados requisitada para os métodos simulados.

Os piores resultados em termos de *outage* foram os métodos que consideram **Apenas SNR** [Lima et al. 2012] e **Único Serviço**, mostrando um resultado próximo de 100%

na maioria os requisitos de taxa. O desempenho do método **Apenas SNR** é facilmente explicado pelo fato de que a interferência é muito degradante em sistemas com pequenas células. Este resultado comprova a inviabilidade em desconsiderar a interferência em cenários mais densos e reforça a importância do estudo do gerenciamento de interferência.

O *outage* do método com apenas um serviço pode ser explicado pelo fato do mesmo não fazer distinções entre borda e centro da célula, sendo assim, existe uma tendência em satisfazer os usuários mais próximos do transmissor. Mesmo que o sistema tenha recursos suficientes para satisfazer todos os usuários, o objetivo de maximização da soma da taxa total fará que após a solução conseguir satisfazer as restrições mínima QoS, o sistema aloque recursos livres para melhores grupos que possuem as melhores SINR, pois estes grupos levarão a um maior ganho de taxa na soma total. Neste caso é esperado soluções com o menor número possível de usuários de borda satisfeitos, prejudicando os terminais móveis de borda. Para validar esta hipótese analisamos a média de usuários de centro de célula e borda satisfeitos em todo sistema considerando as soluções ótimas e **Único Serviço** em todos os TTIs. A Figura 3 contém a média de usuários satisfeitos para centro e borda em todo sistema para o maior requisito de taxa simulado, 100 kbps.

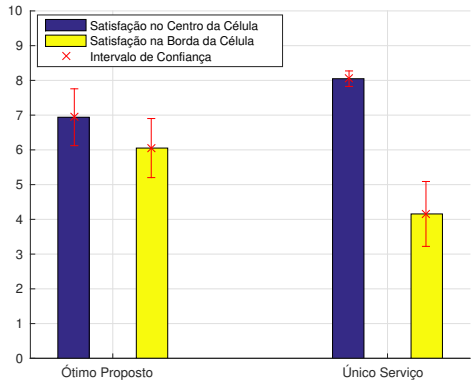


Figure 3. Média de usuários satisfeitos para requisito de taxa de 100 kbps.

Para este resultado consideramos os TTIs em que os métodos encontraram alguma solução. No caso do método **Único Serviço**, consideramos os TTIs em que há 4 usuários satisfeitos em cada célula, mesmo que a solução não tenha atingindo o número mínimo de usuários satisfeitos na borda e no centro da célula, o que contribui para a taxa de *outage* mostrado anteriormente. É importante destacar que a solução ótima proposta satisfez em média 7 usuários de centro e 6 de borda de célula. No total o sistema simulado possui 9 usuários em cada caso, sendo necessário satisfazer os 6 móveis obtidos na solução ótima. Em contra partida um desbalanceamento entre a satisfação dos usuários é verificada na solução com **Único Serviço**. Em média as soluções apresentaram satisfação em 8 usuários do centro contra 4 da borda, ferindo o requisito de justiça adotado. Este resultado evidencia a falta de justiça da solução sem requisitos de taxa próprios para usuários de borda, o que prejudica a experiência uniforme dos usuários na cobertura do sistema.

6. Conclusões

A densificação é um dos principais métodos para ganhos de capacidade em sistemas de telecomunicações e é considerada uma das principais linhas de pesquisa para a quinta geração de comunicações móveis. O aumento da densidade de Estação Rádio Base (ERB)

além de permitir um maior reuso de recursos de rádios no sistema também implica no aumento da interferência que se torna um limitante. Técnicas de gerenciamento de interferência vêm sendo estudados na literatura para diversos objetivos como, aumento da soma de taxa total do sistema, maximização da mínima SINR (do inglês, *Signal to Interference plus Noise Ratio*), entre outros.

Uma forma de mitigar a interferência é administrar o reuso de recursos em cada célula de forma a explorar os ganhos dos canais de interesse e interferente eliminando (ou diminuindo) assim a interferência. Neste contexto este trabalho propõe uma solução ótima para alocação de recursos com o objetivo de maximizar a taxa de dados em um sistema multisserviços, considerando critérios de QoS (do inglês, *Quality of Service*) em termos de taxa mínima por serviço e número mínimo de usuários atendidos pelo sistema mantendo justiça entre usuários de centro de borda das células. O problema proposto foi linearizado na forma de um ILP (do inglês, *Integer Linear Program*) e transformado em uma versão compacta que possibilita o uso de *softwares* comerciais para a solução. Não foi encontrada na literatura até o momento alguma proposta com estas características para cenário com multicelulas, tornando a proposta um novo critério de comparação para futuros trabalhos relacionados.

Apesar da solução ótima do problema ser possível utilizando BB (do inglês, *Branch-and-Bound*), a complexidade computacional da solução é exponencial, o que dificulta sua aplicação em cenários com número maior de usuários. Uma perspectiva de trabalho futuro é a proposta de uma solução heurística de baixa complexidade para o problema.

Agradecimento

Este trabalho foi apoiado pelo *Innovation Center Ericsson Telecomunicações S.A.*, Brasil sob contrato de cooperação técnica EDB/UFC.42. O estudante Lászlon R. Costa gostaria de agradecer a Fundação Cearense de Apoio a Pesquisa (FUNCAP) pelo suporte financeiro.

References

3GPP (2009). Evolved Universal Terrestrial Radio Access (E-UTRA). Technical Report TR 36.213, Third Generation Partnership Project.

3GPP (2012). Feasibility study for further advancements for e-utra (lte-advanced). Technical Report TR 36.912 V11.0.0, Third Generation Partnership Project.

Akbari, M. and Vakili, V. (2012). Resource allocation for ofdma systems with trade-off between throughout and quality of service. In *2012 Sixth International Symposium on Telecommunications (IST)*, pages 420–425.

Bhushan, N., Li, J., Malladi, D., Gilmore, R., Brenner, D., Damnjanovic, A., Sukhavasi, R., Patel, C., and Geirhofer, S. (2014). Network densification: the dominant theme for wireless evolution into 5g. *IEEE Communications Magazine*, 52(2):82–89.

Bohge, M. (2010). *Dynamic Resource Allocation in Packet-Oriented Multi-Cell OFDMA Systems*. PhD thesis, Berlin Technology University.

Chu, D. (1972). Polyphase codes with good periodic correlation properties (corresp.). *IEEE Transactions on Information Theory*, 18(4):531–532.

Gotsis, A. G. and Alexiou, A. (2013). On coordinating ultra-dense wireless access networks: Optimization modeling, algorithms and insights. *CoRR*, abs/1312.1577.

Heimiller, R. (1961). Phase shift pulse codes with good periodic correlation properties. *IRE Transactions on Information Theory*, 7(4):254–257.

IBM (2009). IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/. Acessado em 09 de dezembro de 2015.

Jang, J. and Lee, K. B. (2003). Transmit power adaptation for multiuser ofdm systems. *IEEE Journal on Selected Areas in Communications*, 21(2):171–178.

Kuhn, H. W. (2010). The Hungarian Method for the Assignment Problem. In Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A., editors, *50 Years of Integer Programming 1958-2008*, pages 29–47, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lima, F. R. M., Maciel, T. F., Freitas, W. C., and Cavalcanti, F. R. P. (2012). Resource assignment for rate maximization with qos guarantees in multiservice wireless systems. *IEEE Transactions on Vehicular Technology*, 61(3):1318 –1332.

Lopez-Perez, D., Chu, X., Vasilakos, A., and Claussen, H. (2014). Power minimization based resource allocation for interference mitigation in ofdma femtocell networks. *IEEE Journal on Selected Areas in Communications*, 32(2):333–344.

Mohsen, N. and Hassan, K. S. (2015). C-ran simulator: A tool for evaluating 5g cloud-based networks system-level performance. In *IEEE International Conference: Wireless and Mobile Computing, Networking and Communications (WiMob),*, pages 302–309.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA.

Rahman, M. and Yanikomeroglu, H. (2007). Multicell downlink ofdm subchannel allocations using dynamic intercell coordination. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 5220–5225.

Rahman, M. and Yanikomeroglu, H. (2008). Interference avoidance through dynamic downlink ofdma subchannel allocation using intercell coordination. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 1630–1635.

Sadr, S. and Adve, R. (2014). Partially-distributed resource allocation in small-cell networks. *IEEE Transactions on Wireless Communications*, 13(12):6851–6862.

Sadr, S., Anpalagan, A., and Raahemifar, K. (2009). Radio Resource Allocation Algorithms for the Downlink of Multiuser OFDM Communication Systems. *IEEE Communications Surveys Tutorials*, 11(3):92 –106.

Sierksma, G. (2001). *Linear and integer programming: theory and practice*. CRC Press, 2nd edition.

Song, G. and Li, Y. (2005). Cross-layer optimization for ofdm wireless networks-part i: theoretical framework. *IEEE Transactions on Wireless Communications*, 4(2):614–624.

Zhang, Y. J. and Letaief, K. B. (2004). Multiuser Adaptive Subcarrier-and-Bit Allocation with Adaptive Cell Selection for OFDM Systems. *IEEE Transactions on Wireless Communications*, 3(5):1566–1575.

An Energy-aware IoT Gateway, with Continuous Processing of Sensor Data

L.E. Talavera[1], **M. Endler**[1], **S. Colcher**[1]

[1] Department of Informatics – Pontifícia Universidade Católica do Rio de Janeiro
Rio de Janeiro, Brazil

{lrios,endler,colcher}@inf.puc-rio.br

Abstract. *Few studies have investigated and proposed a middleware solution for the Internet of Mobile Things (IoMT), where the smart things (Smart Objects) can be moved, or else can move autonomously, but remain accessible from any other computer over the Internet. In this context, there is a need for energy-efficient gateways to provide connectivity to a great variety of Smart Objects. Proposed solutions have shown that mobile devices (smartphones and tablets) are a good option to become the universal intermediates by providing a connection point to nearby Smart Objects with short-range communication technologies. However, they only focus on the transmission of raw sensor data (obtained from connected Smart Objects) to the cloud where processing (e.g. aggregation) is performed. Internet Communication is a strong battery-draining activity for mobile devices; moreover, bandwidth may not be sufficient when large amounts of information is being received from the Smart Objects. Hence, we argue that some of the processing should be pushed as close as possible to the sources. In this regard, Complex Event Processing (CEP) is often used for real-time processing of heterogeneous data and could be a key technology to be included in the gateways. It allows a way to describe the processing as expressive queries that can be dynamically deployed or removed on-the-fly. Thus, being suitable for applications that have to deal with dynamic adaptation of local processing. This paper describes an extension of a mobile middleware with the inclusion of continuous processing of sensor data, its design and prototype implementation for Android. Experiments have shown that our implementation delivers good reduction in energy and bandwidth consumption.*

1. Introduction

Despite the huge number of potential applications and the increasing proliferation of appliances with embedded processing and wireless communication capacity, yet there is no widely accepted approach, established standards and consolidated technologies for the Internet of Things (IoT) at a global scale. In other words, extraction of meaningful information of data from tens of billions of sensors and actuators in (near) real-time is still a challenge. In particular, very few studies have focused on the Internet of Mobile Things (IoMT) where a Mobile Object (M-OBJ) may be any movable object (*Thing*) that carries sensors and/or actuators and has some means of wireless connectivity. In this context of general and unrestricted mobility of Smart Objects, different resources (e.g. embedded and wearable sensors) can become available and unavailable at any moment without previous warning.

Nowadays, IoT is evolving towards a heterogeneous network including a mix of IP-based connectivity and an array of short-range, wireless technologies (e.g., Bluetooth, NFC, ANT+), the latter used by peripheral devices in the edge networks. Furthermore, according to [daCosta 2013], IPv6 does not solve all IoT problems because management, rather than addressing and routing, is the biggest challenge of IoT.

IoT communication involves small but frequent messages, where each message individually is unimportant, but the statistical properties of the corresponding data flows carry the relevant pieces of information. In this regard, Complex Event Processing (CEP) is a rather novel software technology for continuous real-time processing of high volumes of data items, that allows an easy specification of patterns that represent different situations. CEP processing combines heterogeneous data from different sources to find patterns and generate high-level events (e.g. a crash or a fraud). Nevertheless, current CEP solutions are server centralized imposing a high energy cost for the communication.

Moreover, recent works propose the use of mobile devices (e.g. smartphones and tablets) as the propagator nodes in IoT, since they can gather the information of the environment and transmit it to the cloud. However, the huge volume of transmissions among nodes (mobile devices) and cloud would drain the device's battery and delay the delivery of data if the network quality is not good enough to provide sufficient bandwidth. As the data volume grows, this approach for sole cloud-based processing becomes less suitable [Saleh and Sattler 2013]. For this reasons, many works advocate that it is important to have as much in-network processing as possible, by performing some functions of filtering, aggregation and pre-processing over the data streams before sending any information to the cloud. In fact, recent works has shown that current mobile devices have the sufficient capacity to execute CEP, allowing them to perform such processing [Stipkovic et al. 2013].

In addition, CEP allows the re-configuration of its processing (deploy/un-deploy queries) on-the-fly. This characteristic is very useful for mobile devices since they can be found in different environments with different resources and thus different processing requirements. Our main contribution is to provide a method for dynamic processing of sensor data in the IoT-gateways and show the feasibility of using CEP in mobile devices by suggesting which kind of processing should or shouldn't be executed on them.

The remainder of this paper is structured as follows. In the next section we give an overview of the enabling technologies and main concepts of our approach towards IoMT. Section 3 discusses the energy consumption in mobile devices and its main concerns. In section 4, we present our approach to handle in-network processing and its general architecture. In section 5 we describe and discuss the results of the performance tests. In section 6 related work is discussed. Finally, in section 7, we draw concluding remarks and point to future work.

2. Enabling Technologies

This section presents the main concepts and technologies that form the foundations of our approach.

2.1. The Mobile Hub

The Mobile Hub (M-Hub) is a general-purpose middleware that enables mobile personal devices (smartphones or tablets) to become the propagator nodes (i.e. gateways to the Internet) for the simpler IoT objects or Mobile Objects (M-OBJ) (sensors/actuators) with only short-range WPAN interfaces. This middleware is responsible for discovering and opportunistically connecting many different simple M-OBJs to the Internet in order to be able to "bridge the gap" between the Internet connection to the cloud and the short-range wireless connections established with M-OBJs. Furthermore, the M-Hub provides context information like the current local time and/or the (approximate) location to the data obtained from the M-OBJs to which it is connected [Talavera et al. 2015]. This feature opens up to IoT applications new ways of classifying, filtering or searching data gathered from the M-OBJs. As its first realization it is implemented for Android and uses Bluetooth Low Energy (*BLE*) as WPAN communication technology. BLE is being widely adopted for IoT and allows a low rate and very low-power communications.

2.1.1. Main Components

The M-Hub consists of three services and one manager, all executing in background. The **S2PA Service** is responsible of discovering and monitoring nearby M-OBJs that use the supported WPAN technologies. This service keeps a record of the current provided sensors/actuators (e.g. temperature, accelerometer, humidity) and publish the sensed information to all the components that require it. One of which is the **Connection Service**, where messages are sent to/from the Cloud in a JSON format through an Internet connection. Important messages (e.g. M-OBJ connection/disconnection) are sent immediately to the cloud, while sensor data or low relevance messages (e.g. temperature readings) are grouped, to be transmitted as a bulk message at a certain time interval.

Messages that are going to be sent to the cloud are enriched with context information, like a timestamp and the approximate location. The location is obtained through the **Location Service**, responsible for sampling the M-Hub's current position obtained from different providers like GPS, network, or a manually entered (in case of a fixed location). The periodicity and duration of all of these three service's actions, is influenced by the device's current energy level (LOW, MEDIUM, HIGH), and is set by the **Energy Manager**, which from time to time sample's the device battery level and checks if it is connected to a power source. The communication among all the services is done using an EventBus[1], which is a Publish-Subscribe (PubSub) event bus optimized for Android that helps to decouple the components, and allows the inclusion of different services without many code modifications.

2.1.2. Short-Range Sensor, Presence and Actuation API

One of the main purposes of the M-Hub is to handle M-OBJs with different WPAN technologies (e.g. BLE, ANT+, Classic Bluetooth). To this end, it has a protocol for short-range communication with M-OBJs, based in two interfaces that help developers implement the different technologies uniformly. On the one hand, the *Technology Interface*

[1]http://greenrobot.github.io/EventBus

maps the main capabilities of each WPAN technology to some methods that have to perform the following actions: 1) Discovery of, and connection to M-OBJs, 2) Discovery of services provided by each M-OBJ, 3) Read and write of service attributes (e.g. sensor values, and actuator commands) and 4) Notifications about disconnection of M-OBJs. And on the other hand, the *Technology Listener Interface* is implemented by the S2PA Service to listen to all the important events that occur on the different technologies and publish them for any interested component. The data published has the structure showed in the Figure 1.

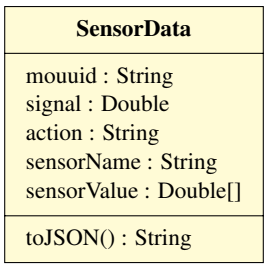


Figure 1. Basic sensor data structure of the M-Hub

The Technologies have to include an ID at programming time to be uniquely identified. This ID is also combined with the M-OBJ's mac address to form the Mobile Object Universally Unique Identifier (MOUUID) which helps developers to differentiate all the M-OBJs, even if they are communicating with the M-Hub under different WPAN technologies. As an example, the module for BLE is defined with the ID 1, and a MOUUID under such technology could be 1-B4994C64BA9F.

2.2. Complex Event Processing

Complex Event Processing (CEP) is a software technology that helps to correlate high volumes of incoming data items which are considered as events happening in the external world such as changes in a company's stock value, or a simple change in the temperature. A CEP solution provides capabilities of filtering, aggregation, correlation and analysis over continuous streams of data. It can detect high-level events that represent different situations and can trigger actions such as notifications or interactions with business processes. In fact, aggregation can be described, as combining and transforming lower level events (e.g. changes in the temperature and humidity) into higher level events (e.g. a fire, a credit-card fraud) to respond upon them as soon as possible.

In CEP, incoming messages runs through a set of continuous queries to produce derived events or sets of data (complex events), this process is performed by an entity called **Event Processing Agent** (EPA). An EPA may perform different kinds of computation on events, such as filtering and aggregation, in order to detect patterns of events. A set of EPAs and the channels they use to communicate form what is called an Event Processing Network (EPN), that can be distributed among multiple physical networks and computers. The top-level architecture of an EPN consists of three layers: event producers, event consumers and event processing logic, plus the communication channels between the layers. An EPA can act as any of the three roles, an event producer at one moment, a consumer at another, or an event processor for the events that it receives from the sources.

Event Producers Also known as event sources, is the layer in charge of producing or capturing events in the internal or external environment (e.g. sensor values, financial trades) to later be forwarded to the event processing logic layer. Event producers could be a software module, different sensors or even a clock.

Event Processing Logic The event processing logic consists of a CEP engine and continuous queries. The CEP engine is where all the events are analyzed, while queries are a set of patterns that describe the situations of interest presented in the form of combinations of events with causal, temporal, spacial and other relations. Queries are defined in a declarative SQL-like language. An example would be a query that detects when several devices are within a certain distance of each other.

Event Consumers Event consumers receive complex events from the Event processing logic in order to deal with detected situations. Typical event consumers could be applications for visualizing events or the cloud.

There are many flavours technologies and languages for CEP, but we focus on Esper[2], since it is an open-source component available under GNU GPL license and one of the most used CEP engine[3]. Esper events allow a rich domain object representation, since it supports all aspects of object-oriented design as well as dynamic typing, while other CEP technologies force a flat Map-like tuple-set definition of events. It also offers a rich set of parameterizable data windows (expiry policies) while most other engines provide a very small set of very simple rolling, sliding or hopping windows[4]. Moreover, different from other CEP technologies it has been successfully ported to Android with the name of *"Asper"*[Eggum 2014]. This allow us to include it as an additional service for the M-Hub.

Esper uses a declarative Event Processing Language (EPL) that derives many properties from the SQL standard, to define the event processing rules used to detect the patterns over the streaming data. An event-processing engine analyzes the event streams and executes the matching rules in-memory[Stipkovic et al. 2013]. An application that embeds Esper, can employ one or more engine instances with different configurations and queries. However, since mobile devices still possess limited resources like processing power it is recommended that all EPAs share one Esper instance instead of having an own instance for each agent [Dunkel et al. 2013].

3. Energy Consumption in Mobile Devices

Mobile devices are general-purpose computers that get their power to process from batteries which have a limited amount of energy. Table 1 shows some of their main characteristics. Many of such characteristics evolved at an exponential rate except for the battery capacity, limiting its use to day-to-day tasks.

Different from desktop and laptop computers, where the CPU is likely to be the component that consumes more energy (often more than 60 percent of the total power), in mobile devices there are several components that may consume nearly the same or event more energy than the CPU. For example, an often cited power budget mobile phone

[2]http://www.espertech.com/esper/documentation.php

[3]http://www.espertech.com/esper/faq_esper.php

[4]http://www.espertech.com/esper/faq_esper.php#comparison

[5]Energy Consumption: Modeling and Optimization. 2014

	1995	2000	2005	2010	2015
Cellular Generation	2G	2.5G - 3G	3.5G	Pre-4G	4G
Standard	GSM	GPRS	HSPA	HSPA, LTE	LTE, LTE-A
Downlink (Mb/s)	0.01	0.1	1	10	100
Display pixels (x 1000)	4	16	64	256	1024
Comms modules	-	-	Wi-Fi, Bluetooth	Wi-Fi, Bluetooth	Wi-Fi, Bluetooth LE, RFID
Battery capacity(Wh)	1	2	3	4	5

Table 1. Evolution of mobile phones[5]

streaming a 384 kb/s video has 1.2 W power drain caused by the network interface, 1 W for the display, and 0.8 W for the CPU and memory operations [Tarkoma et al. 2014]. Besides, mobile devices possess several sensors that also consume a considerable amount of energy, such as GPS, accelerometer, and gyroscope. Thus, it is important to understand where and how the energy is used.

Some experiments presented in [Pathak et al. 2012] confirms that most applications spend a large amount of energy in I/O components such as Wi-Fi, 3G and GPS. With the increasing popularity of mobile internet services, wireless data transmission is becoming a major energy consumer on mobile Internet devices [Tarkoma et al. 2014]. For example, according to a review from AnandTech[6], the Motorola Moto X (2013) device can be idle for up to 576 hours, but can only maintain up to five hours of data access on 4G and eight hours on Wi-Fi. In other words, there is a considerable in energy consumption when keeping the mobile device idle (fully awake without active applications) and when it is using the wireless Internet connection [Pentikousis 2010].

Wireless communication interfaces such as Wi-Fi and Bluetooth have a dynamic power demand. They usually have three states: idle, transmitting and receiving which can be categorized in two different duty cycles, idle and active (transmissions and receptions). The power consumption on the active state is significantly higher than on the idle state. The overall power can be optimized by making active states shorter, so mobile devices could spend most of their time in idle states [Tarkoma et al. 2014] consuming less energy.

4. Mobile Processing and Dynamic Deployment

The Mobile Event Processing Agent (MEPA) is an aditional service that extends the Mobile Hub (M-Hub) with the capacitity of local processing of the data received from the Mobile Objects (M-OBJs). Most of the current IoT gateways only care about the direct transmission of data acquired from the physical environment to the cloud[Pereira et al. 2013]. But differently from the traditional Internet where data consumption is primarily discrete[Billet and Issarny 2014], IoT deals with a continuous processing of data produced by sensors and actuators embedded into M-OBJs. Furthermore, communication with the cloud is an expensive operation in terms of energy consumption and network bandwidth which is critical regarding the scalability and the sustainability required by the IoT. Thus, a recurrent IoT concern is to reduce the amount of information being sent to the cloud, by detecting and sending only consolidated and pre-processed data that actually matters to applications, like a sudden increase in the temperature, or an elevated heart rate.

[6]http://www.anandtech.com/show/7235/moto-x-review/6

4.1. MEPA Service

Given the need to reduce bandwidth consumption in order to avoid bottlenecks, IoT should completely decentralize its processing. Nowadays mobile devices have enough processing power to perform part of the IoT data processing close to the data sources (M-OBJs). In this context, we believe that Complex Event Processing (CEP) can help to evaluate streams of sensor data by checking for certain data patterns. Some of these patterns could be their timestamps and order of occurrence expressed as queries (e.g. consecutive temperature events with a value increase). Raw sensor data usually carries only little semantic information, and hence it needs to be correlated and enriched to gain some meaning [Dunkel et al. 2013]. Moreover, CEP exhibits characteristics that makes it well suited for processing in mobile devices: it employs in-memory processing which allows (near) real-time operations and also the ability to correlate heterogeneous data.

We have included Asper CEP engine [Eggum 2014] in the MEPA Service to process any incoming sensor data. It keeps a record of the running CEP rules, and allows to start and stop them on-the-fly. The sensor data that arrives from M-OBJs are defined as a primitive event type, which contains the sensor's names and their respective values (see Figure 1). As shown in the Figure 2, the MEPA Service is subscribed to all the messages that are sent from the S2PA and Connection services, since the former collects the data from the M-OBJs, and the latter receives commands from the cloud to modify the behavior of the MEPA Service itself (e.g. deploy a new CEP rule). Every time an event pattern is detected (which leads to the generation of a new complex event), it will be published to any interested component that could be the Connection service, or another rule in the MEPA Service.

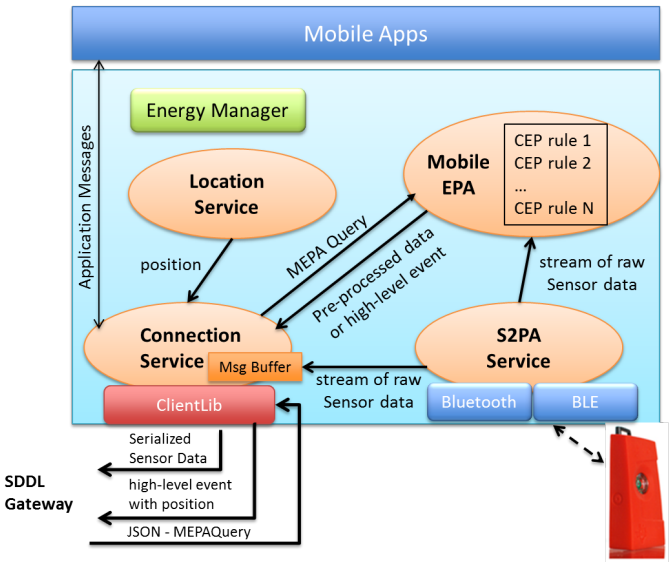


Figure 2. Architecture of the M-Hub with the MEPA Service

CEP rules frequently implement complex correlation functions that sometimes can be divided into different sets of sub-queries. These sub-queries can perform certain steps of the correlation separately and provide the results to a root CEP rule in the cloud. For example, a root CEP rule could be a computational-intensive processing, while the rest of sub-queries could be simple correlations on collected sensor data [Schmidhäuser 2014].

Since mobile devices collect most of the sensor data, processing these data directly in them can improve the performance of the system regarding the possible slow network connections, as well as reducing the device's energy consumption.

4.2. Dynamic Deployment of CEP Rules

M-Hubs can opportunistically collect data in different environments (e.g. a hospital, a school) at different moments. Such environments are very likely to have diverse sets of M-OBJs, which provide sensor data to complete different applications. For example, a M-Hub in a hospital could interact with temperature, heart rate and accelerometer sensors attached to the patients. Thus it could detect events such as irregular heart rates, fever, or even the fall of a person. A completely different scenario could be described for sensors located in the street or a the beach that could help to detect high pollution/radiation levels, or the average environmental temperature. Due to this flexibility requirement we included the capability to add/remove CEP rules to/from the M-Hub remotely using the communication link with the cloud. Depending on the location, CEP rules in the MEPA Service could be swapped to fit the M-Hub's current usage context. A software framework implemented around the MEPA Service translates commands (JSON messages) received through the Connection Service into specific instructions to modify the CEP rules executing in the M-Hub. Such instructions are encapsulated as a ***MEPAQuery*** object that contains information about the type of the request (i.e. add/remove/start/stop/clear) and the label used to identify the EPL queries. In the case of an `add` request type, the EPL query will be also included as a string. Wrong rules specifications or any other exception that could happen generates an error message that is sent to the server.

5. Performance Experiments and Results

This section describes the experiments and measurements realized over the CEP processing in mobile devices. We tested a prototype running different kinds of CEP rules to measure the number of actions the M-Hub can perform until the battery level decreases by 1%, as well as the total bandwidth usage. For all the experiments we used a Motorola Moto X handheld (model 2013) running Android 4.4.4. Both Android runtimes were tested, Dalvik and ART[7]. The devices used as M-OBJs were off-the-shelf SensorTags[8], and the WPAN technology used for communication was BLE. BLE in Android limits the simultaneous connections to six devices. The type of WLAN used is IEEE 802.11bgn.

To quickly connect with the M-OBJs and avoid that other components affect the battery life, the M-Hub was configured to not use the Location Service, perform a WPAN scan every three seconds, and the scan duration was set at two seconds. For the experiments without CEP processing, the time interval in between consecutive sending of sensor data to the cloud was set to 100ms. Each test sampled the processing and the cloud communication for one hour with different sets of connected M-OBJs (1, 3, and 6). The average size of the messages was 200 bytes for events, and 300 bytes for sensor data. Finally, some external factors that we couldn't control were present during the experiments, such as some disconnections of the M-OBJs since BLE is still unstable in Android, and some processing or Internet communications from other applications that slightly affected the battery life (e.g. system apps).

[7]ART and Dalvik - https://source.android.com/devices/tech/dalvik/

[8]Texas Instruments CC2541 Sensor Tag - http://www.ti.com/lit/ml/swru324b/swru324b.pdf

Table 2. No Processing - Energy and bandwidth consumption

Dalvik	1	3	6
Bandwidth (kb)	7108	20635	42204
Mean time (s)	578.00	512.57	536.17
Std. Deviation	34.77	52.32	7.84
ART	1	3	6
Bandwidth (kb)	7262	19551	37440
Mean time (s)	727.00	629.39	551.60
Std. Deviation	21.85	48.18	13.35

5.1. Filtering Rule

We tested how much energy and bandwidth we could save with a simple CEP filtering rule. Hence, we created a rule that filtered the information to just temperature data, but didn't do any further processing. By filtering data we can successfully reduce the amount of transferred information, reducing the possibility of a bottleneck. However, it doesn't reduce the amount of active states of the network interfaces. As we can see in Table 3, even when we reduced the total bandwidth usage, the energy consumption is almost the same as sending all the information to the cloud (see Table 2). Thus, if we can't reduce the number of active states for communication, we won't be able to see big reductions in the energy consumption.

Table 3. Filtering rule - Energy and bandwidth consumption

Dalvik	1	3	6
Bandwidth (kb)	3146	7178	12682
Mean time (s)	593.60	533.72	488.85
Std. Deviation	39.59	38.72	31.00
ART	1	3	6
Bandwidth (kb)	3210	7045	12865
Mean time (s)	670.40	593.00	530.00
Std. Deviation	32.12	30.36	21.48

5.2. Aggregation Rules

In this experiment we intend to show the energy and bandwidth consumption (see Table 4) with some rules that included other processing than just filtering, and used different CEP window types (i.e. sliding and jumping). The scope of a stream is called a *window* and defines the lower and upper bounds of the information that is currently seen. Jumping windows wait until their size n is filled with events to process in a bulk operation all the data items contained.

Sliding windows have lower and upper bounds that advance with time or as data items are inserted into the engine. Every time these windows vary, they process all the data that they contain within their bounds. Their main difference is that jumping windows will never contain a data item that could have resided in the window before it, while sliding windows only remove the oldest events [Eggum 2014].

Both window types can be classified as time- and count- based. Time-based windows were used for the tests with one minute of length. The CEP rules filter the data to process their average value. Only events that contained the average value were sent to the cloud. The results show that sliding windows don't reduce the active states of the network interfaces. However, jumping windows where the frequency of outbound events depends on the window size, hence bigger windows will decrease the number of active states.

Table 4. Aggregation rules - Energy and bandwidth consumption

	Dalvik			**ART**		
Jumping	1	3	6	1	3	6
Bandwidth (kb)	71	102	69	81	105	67
Mean time (s)	807.50	760.67	719.80	918.00	840.67	824.25
Std. Deviation	44.58	19.01	34.50	26.89	21.55	22.43
Sliding	1	3	6	1	3	6
Bandwidth (kb)	2519	6131	11355	3261	5485	10240
Mean time (s)	576.33	553.83	490.14	705.67	607.20	582.00
Std. Deviation	49.31	45.93	26.87	27.43	32.76	26.28

5.3. Rules with Heavy Processing

In the following experiments we tested a more complex processing, in which the magnitude of the accelerometer sensors [9] was calculated in a window of time. The accelerometer used can measure acceleration in three directions simultaneously. The magnitude of the accelerometer was defined as $||a|| = \sqrt{a_0^2 + a_1^2 + a_2^2}$. Similar configurations as the previous experiment were used.

Table 5. Heavy processing rules - Energy and bandwidth consumption

	Dalvik			**ART**		
Jumping	1	3	6	1	3	6
Bandwidth (kb)	78	148	62	74	105	67
Mean time (s)	721.20	672.80	649.20	951.46	862.67	728.5
Std. Deviation	53.10	57.05	31.73	120.97	40.46	29.05
Sliding	1	3	6	1	3	6
Bandwidth (kb)	3026	4933	10630	2472	5330	10240
Mean time (s)	691.00	519.00	480.14	682.00	566.60	491.20
Std. Deviation	32.79	18.65	31.14	53.76	64.65	11.61

Rules with a more complex processing can consume a reasonable amount of energy depending on their outbound frequency. Hence, using complex CEP queries in mobile devices could be counterproductive if the events have a high rate of occurrence. Nevertheless, queries like the ones presented in the former experiments (Table 5) can still be executed if the frequency of outbound events is low enough to compensate the energy consumption for the processing.

[9]The accelerometer is a device that measures the acceleration in a specific direction from gravity and movement.

5.4. Two Rules (Aggregation and Pattern Match)

Here we tested two different rules executing at the same time. The first one processed the average temperature in a jumping window of 10 seconds, while the second expected four consecutive temperature values, each one higher than the previous and the first one higher than 20. In the case of the second rule we don't have control over the number of outbound events, since it depends on the sequence of temperature events that fulfill the pattern. This experiment is intended to show the energy consumption with two rules executing at the same time. Table 6 presents the energy and bandwidth measurements. Results show that even if we have two rules, if they have a low rate of outbound events, it is still possible to reduce the energy consumption in comparison with Table 2.

Table 6. Two rules - Energy and bandwidth consumption

Dalvik	1	3	6
Bandwidth (kb)	258	389	298
Mean time (s)	750.25	733.25	789.50
Std. Deviation	26.86	20.22	52.67
ART	1	3	6
Bandwidth (kb)	270	326	264
Mean time (s)	895.00	814.67	867.00
Std. Deviation	16.09	37.00	10.44

5.5. Discussion

The experiments show that in most of the cases the MEPA service can significantly decrease the energy and bandwidth consumption compared with sending all the data to the cloud. Moreover, the results indicate that CEP jumping windows rules are better suited to be executed on mobile devices than sliding windows, since sliding windows impose a significant use of CPU, and have a high rate of outbound events. If the data produced by sliding windows is directly send to the cloud, it will generate a significant use of the Internet connection.

Nevertheless, it is also possible to use the output of the CEP rules that use sliding windows as the input for another rule X in the MEPA Service. In such case, if the rule X is the one that communicates with the cloud and can significantly reduce the outbound frequency, energy consumption can still be reduced. Moreover, the bigger the size of the jumping windows, the more energy consumption that can be reduced. Since it will allow to increase the time the network interface remains in idle state. In the case of pattern match rules, we have no control over the amount of generated events, thus the expected frequency of the pattern will determine if they are suitable as mobile processing.

We can conclude that the impact of having one or more rules depends on the frequency of detected events. Hence, local processing should as much as possible reduce the frequency and the number of message transmissions to the cloud. However, if the complexity of the processing is high, and the probing of the sensor data transmission is infrequent, then sending them to the cloud could be better (but only as bulk messages with many sensor data). In the results we can also see that using ART the decrease of energy is mitigated. The use of ART also resolves many performance related issues that was previously seen with Dalvik[Eggum 2014].

6. Related Work

Several works in both industry and academia propose the use of average Things (moderately powerful Things) as the enablers of the Internet of Things (IoT). These devices could act as temporary IP routers and opportunistic context providers for simpler Things. Nevertheless, only recent works concern about the amount of transferred data and energy consumption on the IoT-gateways that are usually energy-constrained devices. [Billet and Issarny 2014] let the sensor networks perform as much in-network processing as possible before sending any data to a proxy or the cloud. However, in-network processing should be restricted to low-latency operations and depends on the available resources in the current location of the IoT-gateways.

[Stipkovic et al. 2013, Dunkel et al. 2013] propose the use of Complex Event Processing (CEP) directly in mobile devices, and avoids the use of the cloud. They reduce transmission traffic, save resources, and private data remains only on the mobile devices (e.g. GPS). However, it can't handle situations where a global view of the data is required, for example aggregation of data obtained from different mobile devices in different geographic locations. [Govindarajan et al. 2014] introduce a new approach where CEP processing is distributed among edge nodes (e.g. wireless sensors, smartphones) and the Cloud (VMs). The pipeline of queries is represented by a graph, where the vertices are the sets of queries, and the edges the event streams that connect the output of a query to the input of the next query. Their proposal only covers the architecture and representation of the queries. They don't consider a dynamic deployment of CEP rules, nor have any evaluation (energy/bandwidth) over the system.

[Chen et al. 2014] propose an architecture for distributed CEP to meet the needs of real-time streaming of information processing. Their approach is based on static environments where gateways are always connected to the same devices. It doesn't provide support for mobile IoT, where gateways can be moved to different environments (e.g. hospital, university), and thus be connected with different sets of sensors. [Stojanovic et al. 2014] propose a system that provides an adaptation of CEP Rules. To achieve this, they propose a context-aware (resources, situation) system, which will dynamically deploy/un-deploy rules to the mobile devices or server. By doing this, better recommendations will be send, better actions will be taken and battery will be saved, since it won't be necessary to have unused rules running on the mobile devices. The problem is that they still send all the information to the server in order to increase their complex knowledge (historical data), moreover they only cares about situations related to health and fitness.

In the work of [Kim et al. 2009], they make use of the DDS (Data Distribution Service) to exchange data/events (sensor data) among nodes, and use CEP (Complex Event Processing) to create useful information to the users. First data is collected from various devices (publishers, e.g. sensors, gps, cameras) and transmitted to each user (subscribers, e.g. pda, smartphone) through the DDS network. The data is processed in each mobile device depending on the user's demands and delivered to the different applications. CEP rules can be changed depending on the requirements of the applications. Nevertheless, they don't consider that the mobile devices can act as data sources and node processors at the same time. Saleh et al. [Saleh and Sattler 2013] solution proposes to distribute CEP rules to the network as a graph, where each node will communicate with each other using

pub/sub. A sensor catalog makes possible to find information about the different sensor devices, such as the computational power, memory size, communication cost and available sensors that will be used at the moment of decide which CEP rules will go to which nodes, or to the server.

Although there are many approaches that try to include CEP in mobile devices, they don't explore its capacity for re-configuration. Moreover, none of the previous works provides a benchmark about the energy/bandwidth consumption of the use of CEP in mobile devices, which are some of the most important aspects to consider when including processing in IoT sensor-gateways.

7. Conclusion

We have presented an IoMT sensor gateway with dynamic local processing of sensor data using conventional smartphones. In fact, the popularity and characteristics of CEP, such as its on-the-fly reconfiguration and fast detection of events, led us to use it for the local processing on mobile devices. Using a prototype implementation and BLE SensorTag devices we did performance experiments that measured and compared the energy and bandwidth consumption between sending all the sensor data, and only pre-processed data to the cloud. The results obtained from these initial experiments are quite encouraging and show that CEP rules that represent events with a low frequency of occurrence are more adequate for mobile devices.

In spite of the encouraging preliminary results, we are aware that our current prototype is only "scratching the surface of IoMT", and much interesting research, software development and applications can be derived from this work. In particular, our future work includes: a way for balancing the CEP processing among cloud and mobile nodes, investigate the problems and possible approaches inter- M-Hub handover protocols aiming to deliver detected events, in case a M-Hub is unable to establish an Internet connection. Moreover, we also plan to study means of sending commands to M-OBJs with actuators, and thus support any Internet-wide remote control of smart things, such as home appliances where an event can start an action locally without the need to sent any information to the cloud.

Acknowledgements

The Mobile Hub project is supported by the *PUC-Rio Microsoft Open Source Alliance*, and partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

References

Billet, B. and Issarny, V. (2014). Dioptase: a distributed data streaming middleware for the future web of things. volume 5. Springer London.

Chen, C. Y., Fu, J. H., Sung, T., Wang, P.-F., Jou, E., and Feng, M.-W. (2014). Complex event processing for the internet of things and its applications. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 1144–1149.

daCosta, F. (2013). *Rethinking the Internet of Things*. Apress.

Dunkel, J., Bruns, R., and Stipkovic, S. (2013). Event-based smartphone sensor processing for ambient assisted living. In *Autonomous Decentralized Systems (ISADS), 2013 IEEE Eleventh International Symposium on*, pages 1–6.

Eggum, M. (2014). *Smartphone Assisted Complex Event Processing*. dissertation, University of Oslo.

Govindarajan, N., Simmhan, Y., Jamadagni, N., and Misra, P. (2014). Event processing across edge and the cloud for internet of things applications. In *Proceedings of the 20th International Conference on Management of Data*, pages 101–104. Computer Society of India.

Kim, D., Lee, J. H., Cheol, R., Jo, J. C., and You, Y. D. (2009). Embedded cep engine used in dds-based mobile devices for differentiated services for customers. In *Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on*, pages 645–646.

Pathak, A., Hu, Y. C., and Zhang, M. (2012). Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems*, pages 29–42.

Pentikousis, K. (2010). In search of energy-efficient mobile networking. In *Communications Magazine, IEEE*, pages 95–103.

Pereira, P. P., Eliasson, J., Kyusakov, R., Delsing, J., Raayatinezhad, A., and Johansson, M. (2013). Enabling cloud connectivity for mobile internet of things applications. In *Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, SOSE '13, pages 518–526, Washington, DC, USA. IEEE Computer Society.

Saleh, O. and Sattler, K.-U. (2013). Distributed complex event processing in sensor networks. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 2, pages 23–26.

Schmidhäuser, S. (2014). *Dynamic Operator Splitting in Mobile CEP Scenarios*. PhD thesis, University of Stuttgart.

Stipkovic, S., Bruns, R., and Dunkel, J. (2013). Pervasive computing by mobile complex event processing. In *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, pages 318–323.

Stojanovic, N., Stojanovic, L., Xu, Y., and Stajic, B. (2014). Mobile cep in real-time big data processing: Challenges and opportunities. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pages 256–265. ACM.

Talavera, L., Endler, M., Vasconcelos, I., Vasconcelos, R., Cunha, M., and Da Silva E.Silva, F. (2015). The mobile hub concept: Enabling applications for the internet of mobile things. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 123–128.

Tarkoma, S., M., S., E., L., and Y., X. (2014). In *Smartphone Energy Consumption: Modeling and Optimization*.

Trilha Principal do SBRC 2016
Sessão Técnica 26
Medição e Avaliação

Medição de desempenho de rede em *hardware*

Racyus Pacífico[1], Pablo Goulart[2], Ítalo Cunha[2], Marcos A. M. Vieira[2], Dorgival Guedes[2], Alex B. Vieira[3], José Augusto M. Nacif[1]

[1]Universidade Federal de Viçosa – Florestal, MG – Brasil

[2]Universidade Federal de Minas Gerais – Belo Horizonte, MG – Brasil

[3]Universidade Federal de Juiz de Fora – Juiz de Fora, MG – Brasil

{racyus.pacifico, jnacif}@ufv.br, alex.borges@ufjf.edu.br

{pgoulart, cunha, mmviera, dorgival}@dcc.ufmg.br

Abstract. *Measurement and tracking have crucial roles in Software-Defined Networks (SDNs). Unfortunately, most of these techniques are implemented in software at network end-hosts. This approach generates imprecision, high costs, and makes monitoring more difficult. In this paper, we extend Stanford's OpenFlow switch to implement a measurement architecture for SDN networks. Our architecture performs measurements in a simple and scalable way without depending on end-hosts. It allows monitoring the performance at the granularity of flows. We prototype our architecture on the NetFPGA platform. As an initial case study, we implemented a module to measure packet interarrival times and evaluated it in a the packet interarrival time in a realistic testbed. Our results show that our architecture presents a low relative difference (close to 0%) compared to measurement performed in software at end-hosts.*

Resumo. *Tarefas de monitoramento e medição têm papel crucial em Redes Definidas por Software (SDNs). Infelizmente, a implementação de grande parte destas técnicas é realizada por software nas extremidades da rede (nós hospedeiros). Tal abordagem gera imprecisão, alto custo e dificulta o monitoramento. Neste artigo, nós estendemos o comutador OpenFlow de Stanford para implementar uma arquitetura de medições em redes SDN. Esta arquitetura pode realizar medições das redes independentemente dos nós hospedeiros de maneira simples, escalável e barata. Além disso, ela permite observar o desempenho na granularidade de fluxos. Nosso protótipo foi implementado sobre a plataforma NetFPGA. Como instância inicial, nós implementamos um módulo para medir o tempo entre chegada de pacotes em um ambiente real. Nossos resultados mostram que nossa arquitetura apresenta baixa diferença relativa (próxima de 0%) em relação às medições do software nos nós hospedeiros.*

1. Introdução

O monitoramento do tráfego é indispensável para a utilização de mecanismos de engenharia de tráfego nas redes modernas [Daniel et al. 2003]. Desde a década passada o monitoramento de tráfego é um campo ativo em pesquisas na área de redes de computadores devido à dificuldade em se medir grandes volumes de tráfego de maneira exata por fluxo, além da complexidade no desenvolvimento de estruturas de medição. Tarefas de medição podem ser implementadas utilizando contadores de fluxo,

estruturas de dados *hash* e programação de pequenos fragmentos de código adicionado na CPU do comutador [Moshref et al. 2013]. As tarefas de medição baseadas em contadores de fluxo consumem muitos recursos (largura de banda e CPU) em virtude da demanda do monitoramento e de granularidade. Outros tipos de técnicas como estruturas de dados *hash* e programas de medição na CPU necessitam de grande investimento no desenvolvimento do *hardware* e na configuração dos programas de medição [Van Adrichem et al. 2014].

O paradigma SDN surge como uma arquitetura de rede emergente que tem o objetivo de facilitar o gerenciamento da rede e tarefas de medição de maneira programável. Esse paradigma consiste na separação do plano de dados do plano de controle [ONF 2012]. O plano de controle tem como funcionalidade gerenciar os dispositivos da rede por *software* de modo centralizado e o plano de dados apenas encaminhar pacotes de acordo com a ação enviada pelo controlador [Naous et al. 2008]. Utilizando a estrutura de SDN, as tarefas de medição se tornam mais baratas, pois não é necessário grande investimento no *hardware* ou de configuração na rede. Um exemplo de aplicação SDN é o comutador *OpenFlow*. Esse comutador utiliza contadores de pacotes e de *bytes* por fluxo trafegado. O conceito de fluxo pode ser redefinido dinamicamente de acordo com os requisitos da aplicação [Van Adrichem et al. 2014].

O objetivo desse trabalho é propor uma modificação do comutador *OpenFlow* [NetFPGA 2015c] para medir o tempo de chegada entre pacotes dos fluxos trafegados. Essa métrica do tempo entre pacotes foi implementada no *hardware NetFPGA* 1G, permitindo alto poder de processamento de pacotes da rede e medição de fluxos TCP de maneira acurada. Nossa arquitetura pode ser vista como um concentrador de rede que realiza medições dos fluxos trafegados em *hardware* e, assim, dispensa medições em *software* nos nós hospedeiros da rede. O protótipo foi testado numa rede real. Atrasos variáveis pré-definidos foram inseridos no envio de pacotes e comparados com os atrasos medidos por *software*. Todos os testes foram realizados com o propósito de validar o comutador com suporte à medição do tempo de chegada entre pacotes.

As principais contribuições deste trabalho são: (i) definição da arquitetura de medição distribuída de métricas de desempenho de rede sem a necessidade de instrumentação dos nós hospedeiros; (ii) implementação da arquitetura proposta em um protótipo que utiliza um comutador SDN de código aberto executando na plataforma *NetFPGA*. Os resultados obtidos mostram que as medições realizadas no nosso protótipo apresentam diferenças relativas próximas de 0% em relação às medições em *software*. Ressaltamos que o estudo é inédito e que outras abordagens realizam medições aproximadas ou apenas em *software*, não sendo possível uma comparação direta. Portanto, comparamos com o caso tradicional, medição fim a fim com a geração de carga na origem.

O artigo tem a seguinte organização: Na seção 2 são apresentadas as principais técnicas de medição em SDN. Em seguida (seção 3) são abordados os detalhes da arquitetura proposta e da implementação do comutador *OpenFlow* modificado na *NetFPGA*. Na seção 4 é descrito o ambiente de testes e a metodologia. Na seção 5 são apresentados os resultados dos testes reais. Na seção 6 são abordados os trabalhos relacionados sobre métricas de medição, técnicas e ferramentas para realizar medições em redes. Finalmente, na seção 7 apresentamos nossas conclusões e trabalhos futuros.

2. Medição de Desempenho de Rede em SDN

Desde a década de 1970 as tecnologias de redes têm sofrido grandes modificações, tornando o gerenciamento das redes uma tarefa complexa e contribuindo para o aumento do custo operacional. Essas dificuldades incentivaram os pesquisadores a desenvolverem novas tecnologias, serviços e dispositivos para prover uma forma fácil e barata para gerenciar essas redes. As redes SDN surgiram para preencher esta lacuna no gerenciamento de redes, além de flexibilizar o roteamento de pacotes. Essa tecnologia permite gerenciar ambientes distintos de redes que englobam dispositivos móveis, *data centers*, serviços em nuvem, virtualização [Casado et al. 2012].

Nas redes SDN, o plano de dados é separado do plano de controle. As funções de encaminhamento (plano de dados) e roteamento (plano de controle) dos pacotes são acopladas nos roteadores tradicionais. O plano de dados das redes SDN é composto por tabelas de fluxo. O armazenamento da entrada de fluxo é feito na memória TCAM (*Ternary Content-Addressable Memory*) do comutador. Esse tipo de memória permite a inserção de regras com agregador * (*wildcard*). O plano de dados na arquitetura SDN encaminha os pacotes com base nas regras de fluxo inseridas nas tabelas de fluxo no comutador [Naous et al. 2008]. Ele também pode ser responsável por monitorar os fluxos e coletar estatísticas [Braun and Menth 2014]. O plano de controle é composto por controladores que são totalmente programáveis e permitem administradores de rede definir roteamento de pacotes por fluxo. Toda a complexidade da rede é colocada no controlador central. Esse controle centralizado possibilita aos administradores gerenciarem recursos da rede e decidirem quais fluxos devem ser monitorados. Isso habilita a experimentação de novos protocolos em redes utilizando ambientes reais, pois permite que pesquisadores separem o tráfego real do tráfego experimental [Naous et al. 2008].

Acoplar funcionalidades de medição em redes SDN é uma tarefa que tem interessado a comunidade acadêmica. A medição de tráfego em redes SDN consiste em coletar estatísticas dos fluxos em diversos níveis de granularidade, satisfazendo diferentes tipos de aplicação e maximizando a utilização dos recursos da rede [Yassine et al. 2015]. Um exemplo seria detectar mudanças no tráfego dos fluxos agregados na escala de minutos para realizar engenharia de tráfego em redes de grande escala ou identificar mudanças no tráfego em escala de milisegundos para reduzir a latência em *datacenters* [Moshref et al. 2013].

As técnicas de medição podem ser classificadas como passivas ou ativas. Técnicas passivas são conhecidas por monitorar o tráfego da rede sem injetar novo tráfego ou afetar o tráfego existente. Esse tipo de técnica realiza medições locais e sua principal desvantagem está nos custos de instalação dos monitores de tráfego na rede. [Van Adrichem et al. 2014]. Técnicas de medição ativas, por outro lado, enviam tráfego de teste (sondas) para estimar as características de utilização da largura de banda da rede. As sondas podem afetar o desempenho da rede, o que interfere na acurácia das medições [Prasad et al. 2003].

Recentemente, diversos trabalhos tratam de problemas decorrentes da medição de desempenho em SDN. Algumas dessas soluções serão apresentadas na seção 6.

3. Arquitetura Proposta – Instrumentação do Plano de Dados

O comutador *OpenFlow* utilizado neste trabalho é simples, de código aberto e projetado para ser executado na plataforma *NetFPGA*. A estrutura SDN utilizada por este comutador transfere a complexidade dos nós da rede para um ponto central de controle, contribuindo na detecção de falhas e qualidade da medição. Com essa estrutura é possível inserir regras de fluxo para serem medidas diretamente no comutador e capturar o tempo das medições por programas executados no espaço de usuário [Naous et al. 2008].

Neste trabalho modificamos o projeto do comutador *OpenFlow* de *Stanford* [Naous et al. 2008] para medir o tempo de chegada entre pacotes dos fluxos trafegados. Implementamos as medições da média e variância do tempo de chegada entre pacotes. Nossa implementação mede 28 fluxos TCP concorrentemente em intervalos definidos pelo usuário na velocidade da interface (*line-speed*), sem afetar a taxa de processamento de pacotes. A arquitetura também faz medições para todos os pacotes do fluxo, sem amostragem, e escala de acordo com o número de regras suportadas pelo comutador. Todas as regras de fluxo são inseridas diretamente na memória ternária (TCAM) do comutador. A TCAM do *hardware* permite a inserção de 32 regras de fluxos com campos exatos e coringas (prefixos). Destas 32 regras reservamos 28 para medição e quatro para o encaminhamento de pacotes ARP diretamente sem que eles passem pelo controlador. Nosso trabalho abre a possibilidade de realizar medições que não eram possíveis de maneira fim-a-fim, por exemplo, a medição do tempo de chegada entre pacotes de fluxos de uma subrede utilizando regras com agregador * do *OpenFlow*.

O *pipeline* do nosso plano de dados é dividido em quatro estágios: extração do identificador do fluxo do pacote (*Header Parser*), busca do identificador de fluxo na TCAM (*Wildcard*), cálculo das operações de medição (*Measure*) e armazenamento das operações de medição na SRAM (*SRAM*). Todos os estágios, exceto *Measure*, pertecem ao projeto do comutador *OpenFlow*. Esses estágios foram adaptados para dar suporte à medição do tempo de chegada entre pacotes. O plano de controle da arquitetura é dividido duas aplicações: o controlador SDN que insere as regras dos fluxos a serem medidos e o *software* de medição que realiza a leitura das medições processadas pelo *hardware*. Na figura 1 é apresentado a arquitetura do protótipo implementado.

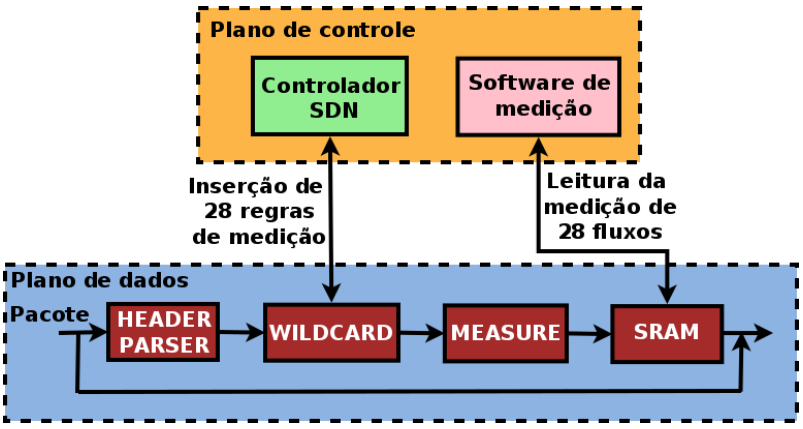


Figura 1. Arquitetura *OpenFlow* com suporte à medição do tempo de chegada entre pacotes.

3.1. Plano de dados

O processo de medição no plano de dados começa com a chegada de um pacote TCP ao módulo que analisa o cabeçalho (*Header Parser*). Este módulo extrai o identificador do fluxo e marca o tempo de chegada do pacote (em ciclos de relógio) gerados pelo módulo contador de tempo (*Time Counter*), e os envia para o módulo de medição (*Measure*). O módulo de medição insere essas entradas em uma fila interna. Este módulo precisa esperar a saída do módulo de busca aproximada (*Wildcard*) para decidir se descarta o tempo de chegada e o identificador do fluxo do pacote ou se calcula a média e a variância. O módulo *Wildcard* compara o identificador do fluxo extraído do pacote com as regras de medição armazenadas na memória do comutador. Após a busca ser completada, o módulo *Measure* envia um sinal (*hit*) indicando se encontrou um identificador de fluxo e a sua posição na memória do comutador.

A partir deste momento iniciam-se os cálculos das operações de medição. A métrica implementada consiste em receber o tempo de chegada do primeiro pacote (T_1) do fluxo (F_j), o tempo de chegada do segundo pacote (T_2) do fluxo (F_j) até o tempo de chegada do enésimo pacote (T_n) do fluxo (F_j). F_j refere-se aos fluxos a serem medidos, tal que, j varia de 1 a 28. A cada intervalo entre pacotes do mesmo fluxo é realizado a operação de diferença $dT = T_c - T_a$, sendo, T_c o tempo de chegada do pacote atual e T_a o tempo de chegada do pacote anterior. Em seguida são realizados os cálculos do acúmulo da soma da diferença, $S_j = S_j + dT$ e o acúmulo da soma ao quadrado da diferença, $S_j^2 = S_j^2 + dT^2$. Por fim, o contador de pacotes fluxo é incrementado (N_j).

Após todos os cálculos terem sido realizados, os resultados obtidos são salvos temporariamente em um *buffer* interno do módulo de medição e na memória SRAM. O *buffer* no módulo de medição tem 28 linhas. Em cada linha é armazenado a quantidade de pacotes por fluxo, o tempo de chegada do último pacote, a soma (S_j) e soma ao quadrado (S_j^2) dos acúmulos dos intervalos dos tempos de chegada dos pacotes. As 28 linhas do *buffer* equivalem às linhas da TCAM às quais são inseridas as regras. Quando ocorre um acerto (*hit*) na busca, o endereço de acerto corresponderá à linha do *buffer* no qual deve ser feita a leitura e a atualização das informações do fluxo do pacote. Por exemplo, se ocorrer um sinal de acerto no endereço cinco da TCAM, as informações da linha cinco do *buffer* são atualizadas. O *buffer* evita que a SRAM seja acessada constantemente. As operações de leitura na SRAM gastam três ciclos de relógio, o que prejudica o desempenho. Isso ocorre devido ao tempo que a máquina de estados espera o dado da memória [Goulart et al. 2015].

Com o resultado das operações salvas na SRAM entra em ação o *software* de medição que é encarregado de terminar a medição e ler via interface de registradores os valores calculados dos fluxos armazenados na SRAM. Devido à limitação do número de células lógicas do *hardware* e a precisão da medição as operações de divisão da média (equação 1) e variância (equação 2) foram realizadas no *software* de medição. Na figura 2 é apresentado o caminho de dados do comutador *OpenFlow* com suporte a média e variância do tempo de chegada entre pacotes. É importante ressaltar que a arquitetura proposta é genérica e permite que outras métricas sejam implementadas, desde que o *hardware* suporte. Um exemplo seria estimar a taxa de transmissão de um fluxo baseado no tamanho do pacote utilizando a média e variância calculada pelo módulo *Measure*.

$$media_j = \frac{\sum_{i=1}^{N_j} dT_i}{(N_j - 1)}, \quad 1 \leq j \leq 28 \tag{1}$$

$$variancia_j = \frac{\sum_{i=1}^{N_j} (dT_i)^2 - (\sum_{i=1}^{N_j} dT_i)^2 / N_j}{(N_j - 1)}, \quad 1 \leq j \leq 28 \tag{2}$$

Atualmente existem modelos de plataformas NetFPGA com FPGAs de maior capacidade, por exemplo, a CML [NetFPGA 2015a] e a SUME [NetFPGA 2015b]. Para efeitos de comparação, o modelo SUME (FPGA Virtex-7) possui 13x mais recursos que o modelo 1G, utilizado nesse trabalho, permitindo monitorar até 416 fluxos concorrentemente.

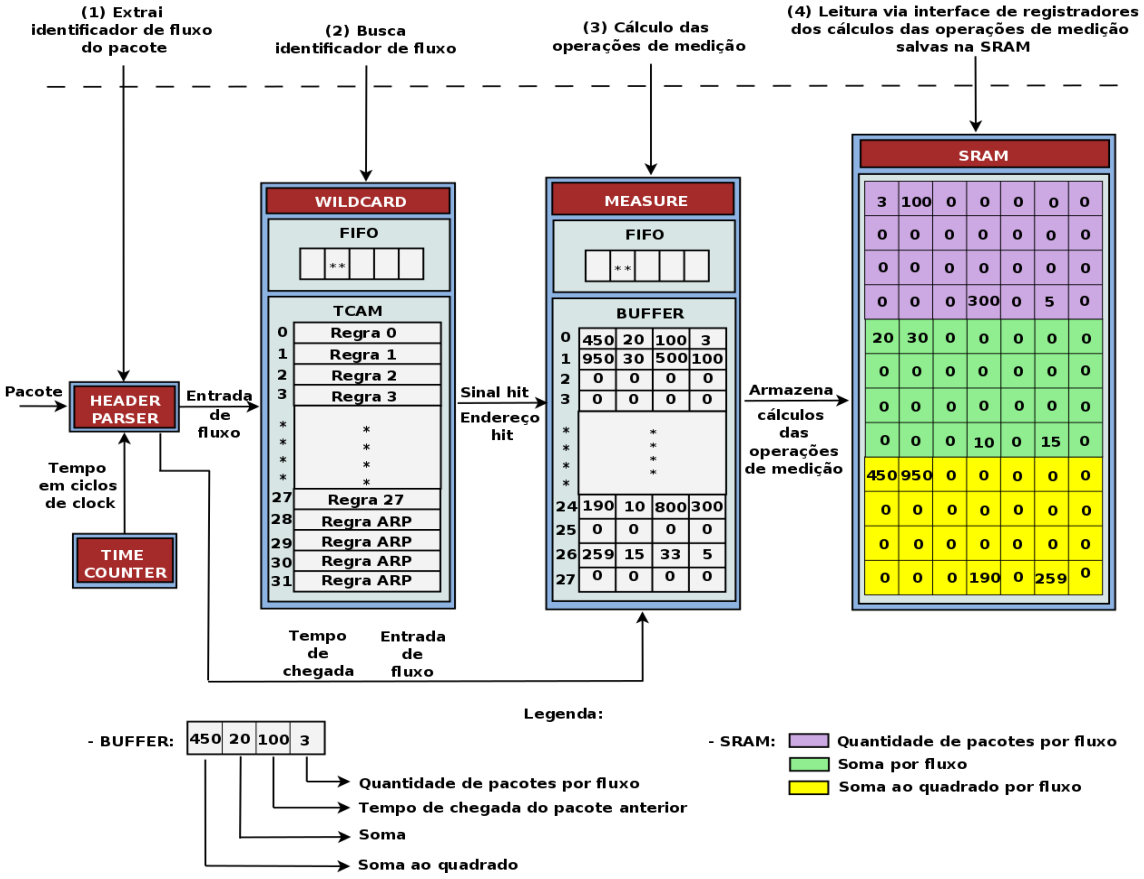


Figura 2. Caminho de dados *OpenFlow* com suporte a média e variância do tempo de chegada entre pacotes.

3.2. Plano de controle

O plano de controle é dividido em duas partes, controlador e *software* de medição. O controlador tem como função inserir na TCAM do comutador as regras que devem ser medidos e dizer ao comutador como os pacotes serão encaminhados. No protótipo desenvolvido o controlador foi programado para medir 28 fluxos TCP e encaminhar os pacotes que pertencem a um fluxo para o terminal B. A especificação das regras dos fluxos a serem medidos são facilmente alteradas. A aplicação do controlador lê um arquivo

com 28 regras e as insere na TCAM. Para modificar as regras dos fluxos que devem ser medidos é necessário apenas alterar o arquivo com as regras e reiniciar o processo responsável pelo controlador. Em SDN existem vários tipos de controladores com suas respectivas características. Utilizamos o controlador POX [OpenNetworkingLab 2015]. Esse controlador é baseado na linguagem *Python* e é utilizado para o desenvolvimento de projetos rápidos de aplicação de rede.

Estendemos o plano de controle criando um *software* de medição. O *software* de medição termina o processo de medição realizando o cálculo resultante da média e variância dos 28 fluxos. Esse *software* lê os valores das operações da média e variância processadas no *hardware* via interface de registradores com base no intervalo definido pelo usuário. A cada leitura, média e variância dos 28 fluxos são calculadas e armazenadas em um arquivo com a data e a hora da medição.

4. Avaliação

Nesta seção apresentamos o ambiente dos experimentos realizados com o protótipo do comutador *OpenFlow* com suporte à medição do tempo de chegada entre pacotes. Nosso objetivo é comparar a acurácia das medições realizadas em *hardware* e *software* com o propósito de validar a medição do protótipo.

4.1. Descrição do ambiente de testes

O cenário de testes consiste em um terminal A enviando pacotes TCP à uma taxa controlada para o terminal B. Os pacotes criados por A não utilizam do protocolo TCP, apenas têm a estrutura do formato do pacote TCP (pacotes TCP RAW). Entre A e B foram inseridos dois comutadores tradicionais contendo o tráfego real com o propósito de aumentar o atraso do tempo de chegada dos pacotes enviados por A e tornar o experimento ainda mais realista. No ponto final da rede foram inseridos a *NetFPGA* rodando o protótipo de medição e o terminal B executando o *software* de medição do tempo de chegada entre pacotes. Em B, a média e a variância são calculadas depois do término do experimento. No protótipo, média e variância são calculadas em tempo real. Nosso objetivo é comparar a diferença relativa entre as medições em B (versão *software*, no hospedeiro) e *hardware* (*NetFPGA*) com base nos atrasos pré-definidos inseridos entre os pacotes enviados por A. Na figura 3 é apresentado o ambiente de testes.

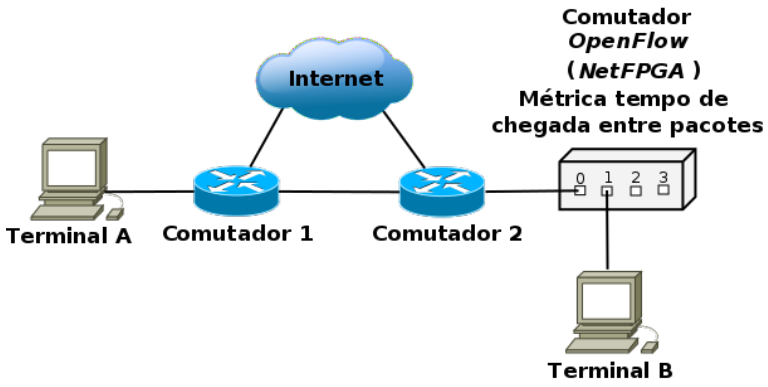


Figura 3. Ambiente de testes.

4.2. Verificação do protótipo

Desenvolvemos um programa para enviar pacotes TCP a partir do terminal A. Esse programa lê um arquivo de texto que contêm as regras de monitoramento definidas no controlador. As regras referem-se aos fluxos monitorados no comutador. O programa executa uma *thread* para cada fluxo lido nesse arquivo. Cada *thread* envia pacotes que correspondem a apenas um fluxo monitorado. Esse programa envia pacotes em intervalos aleatórios ou constantes. Utilizamos intervalos constantes na fase de validação do protótipo para conferir os valores capturados no *hardware*. Para os testes reais, utilizamos intervalos aleatórios para observar os valores medidos. Os pacotes foram gerados utilizando a biblioteca `libnet` do C. Para termos um controle maior do número de pacotes vindos de A para B e *NetFPGA*, escrevemos os pacotes enviados pelo `libnet` em vários arquivos `pcap` de duração de 1s para serem transmitidos por A. Esta abordagem permite uma comparação mais acurada. Arquivos `pcap` em B (*software*) também foram criados e reproduzidos utilizando a biblioteca DPKT da linguagem *Python*.

Foram realizadas várias iterações a partir dos arquivos `pcap` criados no terminal A. Antes de cada execução, o programa `tcpdump` é inicializado em B para capturar os pacotes enviados por A. No fim da iteração, as medições são lidas e a SRAM e o *buffer* da NetFPGA são inicializados. Após essa etapa, o `tcpdump` escreve um arquivo de saída com os pacotes capturados. Esses passos são repetidos em todas as iterações.

4.3. *Warm up*

Os experimentos foram compostos por cinquenta iterações, tendo cada iteração a duração de 1 segundo. São gastos 5 segundos para inicializar o controlador, os processos de monitoramento e a transmissão de pacotes, e 1 segundo para finalizar todos os processos. O tempo total do experimento é o tempo de inicialização somado ao tempo gasto pelas iterações e o tempo da finalização: $5s + 50 \times 1s + 1s = 56s$. Foram descartados 10% das amostras devido ao período de *warm up*, que referem-se à 6 segundos no começo e no fim do experimento. Cada segundo corresponde à 28 amostras.

5. Resultados

Todos os experimentos foram configurados de acordo como descrito na seção 4.1. Os procedimentos descritos nas seções 4.2 e 4.3 foram utilizados na preparação do ambiente. Nesta seção é apresentado os gráficos e intervalos de confiança da média e da variança dos atrasos para cada experimento.

5.1. Experimento 1

Nesse experimento foram enviados aproximadamente 1700 pacotes com atrasos pré-definidos de 20/40 milisegundos (ms) para cada um dos 28 fluxos. Os resultados mostram que 95% da diferença relativa entre as abordagens em *hardware* e *software* estão próximas de 0.0072% no gráfico da média. No gráfico da variância, 95% das medições possuem diferença relativa próximas à 0.26%. Três valores ficaram entre 3% e 7%, mas as diferenças absolutas são menores que 10^{-4}. Os valores esperados para a média estão dentro do intervalo de confiança $10^{-5} \times [2.897 : 3.166]$, e os valores esperados para a variância estão dentro do intervalo de $10^{-3} \times [1.0043 : 1.3574]$.

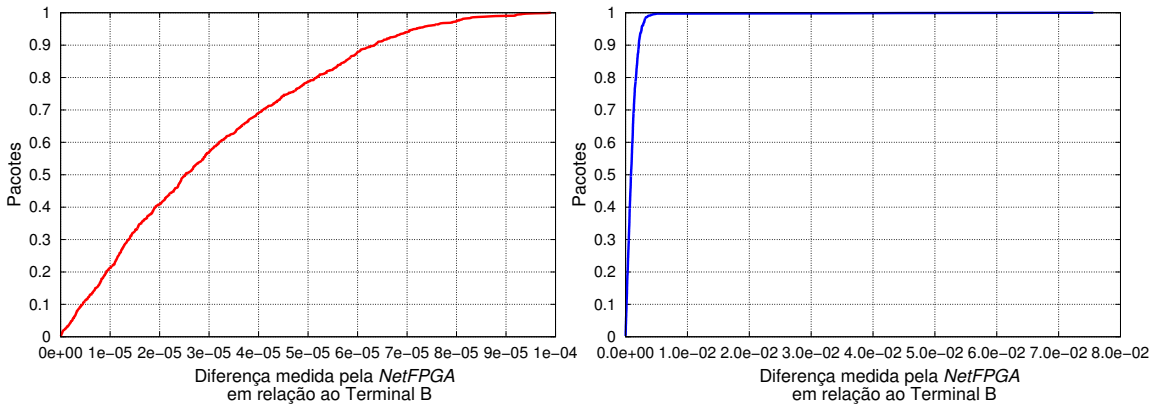


Figura 4. Gráficos CDF diferença relativa média(A) e variância(B) atraso 20/40ms.

5.2. Experimento 2

Nesse experimento foram enviados aproximadamente 1800 pacotes com atrasos aleatórios para cada um dos 28 fluxos. Foram definidas três sementes para gerar os atrasos. O gráfico da média mostra que 95% das medições obtidas com diferença próxima de 0.0071%. No gráfico da variância, 95% das medições obtiveram diferença relativa próximas de 0.19%. Nesse gráfico, seis amostras ficaram entre 3% e 6% de diferença relativa, mas as diferenças absolutas foram menores que 10^{-4}. O intervalo de confiança do gráfico da média é $10^{-5} \times [3.169 : 3.432]$, e os valores esperados da variância estão dentro do intervalo de confiança $10^{-4} \times [7.9642 : 12.4338]$.

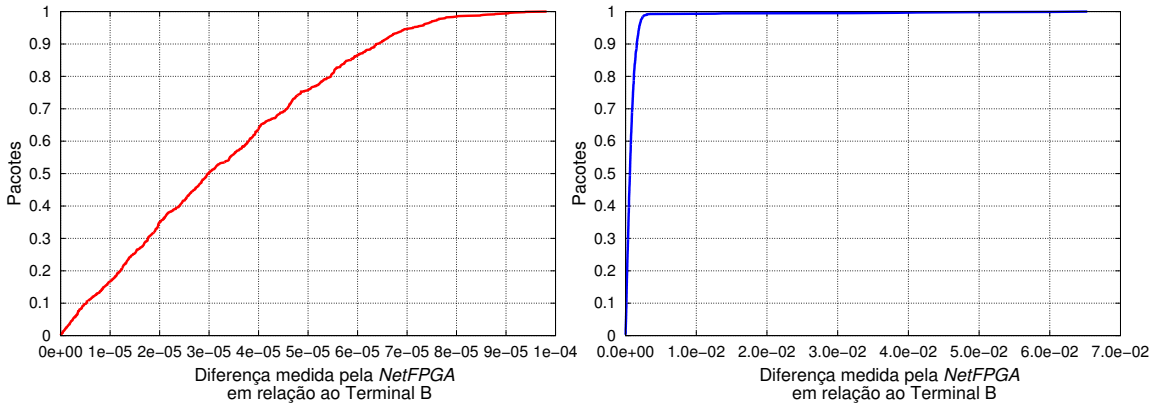


Figura 5. Gráficos CDF diferença relativa média(A) e variância(B) atraso variado.

5.3. Experimento 3

Nesse experimento foram enviados 1700 pacotes com atrasos pré-definidos 20/40ms para quatro fluxos. Objetivo deste experimento foi medir todos esses fluxos numa regra agregada com o propósito de mostrar que a arquitetura permite monitorar o tráfego de subredes inteiras. O gráfico da média mostra que 97% das medições obtidas com diferença próxima de 0.007%. No gráfico da variância, 97% das medições obtiveram diferença relativa próximas de 0.257%. Os valores esperados da média estão dentro do intervalo

de confiança $10^{-5} \times [2.788 : 4.349]$, e os valores esperados da variância estão dentro do intervalo de confiança $10^{-4} \times [8.4223 : 12.072]$.

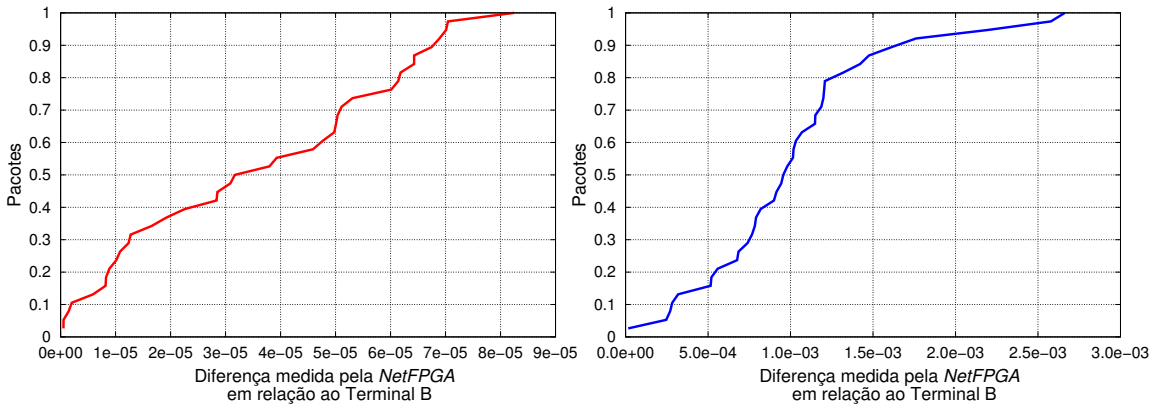


Figura 6. Gráficos CDF diferença relativa média(A) e variância(B) regra wildcard.

6. Trabalhos Relacionados

Nesta seção será apresentado uma visão geral dos trabalhos relacionados com base nas métricas, técnicas e ferramentas utilizadas em medição de redes. Realizamos o levantamento de onze trabalhos de acordo com os seguintes critérios: métricas de medição, a plataforma e o ambiente de implementação.

[Lombardo et al. 2012] apresenta a implementação de quatro módulos que estendem o roteador de referência implementado na *NetFPGA*. Dois desses módulos monitoram a taxa de *bits* nas filas de entrada e saída. Um terceiro módulo realiza a previsão da taxa de bits de entrada a partir de um filtro de médias móveis exponencialmente ponderadas (EWMA), o quarto módulo é uma extensão do limitador de taxa da biblioteca da *NetFPGA*. Um programa no espaço de usuário lê os registradores no *hardware* que armazenam as taxas de *bit* calculadas nas filas de entrada e saída. As diferentes escalas de tempo da amostragem (em microsegundos) e do ciclo de relógio da *NetFPGA* prejudicam a acurácia da medição. O cenário de testes foi composto por duas *NetFPGAs* ligadas entre si: a primeira contendo o projeto do *reference router* com os quatro módulos implementados e a segunda com o projeto *packet generator*, um projeto de referência na *NetFPGA* que gera e transmite pacote a partir de arquivos `pcap`.

No trabalho de [Moshref et al. 2013] é discutido o compromisso entre utilização de recursos da rede com a exatidão das diferentes primitivas de medição. O artigo tem como propósito avaliar a utilização de técnicas baseadas em contadores, estruturas de *hash* e programação da CPU para detectar fluxos pesados (*Hierarquical Heavy Hitters*), que consiste na identificação dos prefixos de IP mais longos cuja utilização da rede supere um limiar pré-estabelecido. Na avaliação do protótipo foram utilizadas várias escalas de tempo e diferentes configurações da utilzação dos recursos dos comutadores. O protótipo foi avaliado utilizando-se *traces* de pacotes obtidos do projeto CAIDA.

Em [Yu et al. 2013] é apresentado a arquitetura de medição OpenSketch. OpenSketch baseia-se na estrutura SDN do comutador *OpenFlow* e utiliza como primitiva

de medição estruturas baseadas a *sketches*. O plano de controle é composto por uma biblioteca de medição enquanto o plano de dados é dividido em três estágios: *hashing*, classificação e contagem. Neste trabalho foram implementados sete *sketches* que possibilitam a medição de cinco diferentes tipos de métricas. A arquitetura OpenSketch foi implementada no *hardware NetFPGA* 1G. A validação deste trabalho ocorreu por meio da simulação de pacotes *trace* CAIDA e pela comparação de duas métricas implementadas no OpenSketch com a amostragem de pacotes do NetFlow.

[Chowdhury et al. 2014] é proposto um *framework* de monitoramento de rede. Esse *framework* fornece uma visão geral e estatísticas sobre a utilização de recurso da rede. O PayLess é construído acima do controlador *OpenFlow* e fornece uma API RESTful para o desenvolvimento de aplicações de monitoramento. A validação do PayLess foi realizada com o monitoramento de utilização de *link* entre comutadores. Toda a topologia foi desenvolvida com base na plataforma Mininet e utilizado o programa *iperf* para o envio de pacotes UDP durante 100s.

[Kekely et al. 2014] apresenta uma implementação de SDM (*Software-Defined Monitoring*) para monitorar o tráfego na camada de aplicação. Os autores implementaram o protótipo sobre o projeto NetCOPE, utilizando FPGAs para trabalhar à taxas de até 100Gb/s. O *hardware* implementado coleta informações do cabeçalho dos pacotes HTTP dos fluxos monitorados e os envia para um programa em espaço de usuário utilizando um formato unificado de pacotes através das interfaces PCI utilizando DMA. As tarefas mais complicadas, como *deep packet inspection* são realizadas em *software* devido as dificuldades em se implementar módulos de análise de pacotes em HDLs. Essa estratégia possibilita alcançar alta vazão no processamento e flexibilidade.

O artigo Planck [Rasley et al. 2014] apresenta uma ferramenta de medição para redes SDN que tem o objetivo identificar as condições da rede com baixa latência. Ambientes como *datacenters* ou de computação em nuvem são caracterizados pela dinâmica do tráfego, o que exige tempos de respostas muito pequenos para controlar o congestionamento ou dar manutenção em falhas na rede. O Planck oferece uma solução eficiente para realizar as tarefas de identificação dos fluxos pelo espelhamento do tráfego numa porta de monitoramento. Como desvantagem, o tráfego através do comutador frequentemente excederá a largura de banda máxima, o que poderá resultar em descartes de pacotes utilizados no monitoramento.

[Van Adrichem et al. 2014] propôs uma implementação de *software* de código aberto para monitorar métricas por fluxo, especialmente, atraso e perda de pacotes em redes *OpenFlow*. Essa implementação fornece engenharia de tráfego ao controlador com medições de monitoramento *online*. A perda de pacotes foi calculada com base nas estatísticas do fluxo do primeiro e último comutador de cada caminho da rede subtraindo o aumento do contador de pacotes do comutador origem com o aumento do contador de pacotes do comutador destino. Essa técnica permite uma medição exata da perda de pacotes. O atraso de pacotes foi calculado por RTT com a injeção de pacotes na rede. Esses pacotes viajaram pela rede e retornaram ao controlador determinando assim os atrasos. Os testes basearam em executar vídeo *stream* entre dois servidores na extremidade da rede ligados por comutadores com atrasos e perda de pacotes estabelecidos com `netem`. Os resultados medidos pelo OpenNetMon foram comparados com *software tcpstat*.

O artigo descreve uma ferramenta de monitoramento de tráfego utilizando técnicas de tomografia de redes para identificar grandes fluxos em redes DCN [Hu and Luo 2015]. O protótipo foi concebido em dois passos. O primeiro passo consistiu na formulação de um modelo de otimização baseado nos valores de uma matriz de tráfego TM preenchida com os valores dos contadores SNMP e SDN dos comutadores, e da definição de uma heurística para solucionar esse modelo eficientemente. O segundo passo consistiu na identificação de grandes fluxos entre os servidores através desses contadores e do modelo de tomografia de rede proposto. A qualidade da ferramenta na escala de *datacenters* foi avaliada por simulações no *ns-3*. Seus resultados mostraram que a identificação utilizando essa abordagem é superior às tradicionais baseadas em contadores.

Em [Mittal et al. 2015] foi implementado um protocolo de controle de congestionamento para *datacenters* que controla a taxa de transmissão usando a técnica de gradiente do RTT. Essa técnica elimina as trocas de contexto do espaço usuário *kernel* contribuindo para uma redução significativa do consumo de CPU e uma baixa latência na rede com alta largura de banda. O gradiente do RTT foi implementado em um *software* rodando sobre uma placa NIC em um sistema operacional com suporte à *OS-bypass*. Foram realizados dois tipos de testes. O primeiro examinou propriedades básicas do controlador de congestionamento tal como: *throughput*, *fairness*, latência e acurácia de tempo. Para o primeiro teste utilizou-se de um pequeno *testbed*. No segundo o protocolo TIMELLY foi executado em um *testbed* de grande escala com 100 máquinas ligadas em uma rede clássica Clos.

[Wellem et al. 2015] aborda uma arquitetura de medição baseada em *sketches* diferente do OpenSketch. Os autores desenvolveram um plano de dados com duas tabelas de fluxo para armazenar identificadores de fluxo e contadores de *bytes*/pacotes sem usar um método de classificação de fluxos que utilize memória TCAM. O protótipo desenvolvido não tem uma estrutura dinâmica, não permitindo a medição de outros tipos de métricas que não sejam baseadas em fluxos. Todo o processamento é realizado no *hardware*, armazenado nas memórias e a medição lida por *software* via interface de registradores. O plano de dados foi implementado na *NetFPGA* 10G e o protótipo validado a partir de pacotes *trace* do CAIDA.

[Zhu et al. 2015] assim como [Mittal et al. 2015] implementou um protocolo de controle de congestionamento para *datacenters*. O protocolo proposto (DCQCN) controla a taxa de transferência dos *hosts* que acessam à memória utilizando RDMA (*Remote Direct Memory Access*). RDMA é um mecanismo de acesso direto à uma memória pré-registrada de outra máquina. Esse mecanismo pode sofrer congestionamento quando a memória compartilhada é acessada por diversos outros *hosts* ao mesmo tempo. A implementação foi totalmente desenvolvida sobre placas NIC contribuindo para redução de consumo de CPU. Os autores validaram o protocolo usando tráfego *benchmark* derivados de tráfegos *trace* de seus próprios *datacenters*.

7. Conclusões

O monitoramento de tráfego é o ponto chave para se garantir a qualidade de redes IPs. As redes SDN e o padrão *OpenFlow* surgem como tecnologias que facilitam as tarefas de gerenciamento e medição dos fluxos. Modificamos o comutador *OpenFlow* de *Stanford* para medir o tempo de chegada entre pacotes dos fluxos trafegados em uma rede realista.

Com base nos resultados obtidos concluímos que a diferenca relativa entre a medição em *hardware* e *software* é quase nula. Isso implica, que a medição do protótipo em *hardware* tem uma acurácia próxima à medida no nodo folha da rede em *software*, com uma maior vazão de pacotes, centralizada e em tempo real.

Como trabalhos futuros, pretendemos implementar as métricas de desempenho tempo de chegada entre pacotes e latência utilizando de estrutura *hash* (*Bloom Filters*) para medir com baixa granularidade o atraso entre pacotes. Pretendemos também implementar outras métricas de desempenho de rede disponíveis na literatura.

Agradecimentos

Este trabalho foi parcialmente financiado por Fapemig, CAPES, CNPq, e pelos projetos MCT/CNPq-InWeb (573871/2008-6), FAPEMIG-PRONEX-MASWeb (APQ-01400-14), e H2020-EUB-2015 EUBra-BIGSEA (EU GA 690116, MCT/RNP/CETIC/Brazil 0650/04).

Referências

Braun, W. and Menth, M. (2014). Software-defined networking using openflow: Protocols, applications and architectural design choices. *Future Internet*, 6(2):302–336.

Casado, M., Koponen, T., Shenker, S., and Tootoonchian, A. (2012). Fabric: a retrospective on evolving sdn. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 85–90. ACM.

Chowdhury, S. R., Bari, M. F., Ahmed, R., and Boutaba, R. (2014). Payless: A low cost network monitoring framework for software defined networks. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–9. IEEE.

Daniel, E. J., White, C. M., Teague, K., et al. (2003). An interarrival delay jitter model using multistructure network delay characteristics for packet networks. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1738–1742. IEEE.

Goulart, P., Cunha, Í., Vieira, M. A., Marcondes, C., and Menotti, R. (2015). Netfpga: Processamento de pacotes em hardware.

Hu, Z. and Luo, J. (2015). Cracking network monitoring in dcns with sdn. In *Proc. IEEE INFOCOM*.

Kekely, L., Pus, V., and Korenek, J. (2014). Software defined monitoring of application protocols. In *INFOCOM, 2014 Proceedings IEEE*, pages 1725–1733. IEEE.

Lombardo, A., Reforgiato, D., Riccobene, V., and Schembra, G. (2012). Netfpga hardware modules for input, output and ewma bit-rate computation. *International Journal of Future Generation Communication and Networking*, 5(2):121–134.

Mittal, R., Dukkipati, N., Blem, E., Wassel, H., Ghobadi, M., Vahdat, A., Wang, Y., Wetherall, D., Zats, D., et al. (2015). Timely: Rtt-based congestion control for the datacenter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 537–550. ACM.

Moshref, M., Yu, M., and Govindan, R. (2013). Resource/accuracy tradeoffs in software-defined measurement. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 73–78. ACM.

Naous, J., Erickson, D., Covington, G. A., Appenzeller, G., and McKeown, N. (2008). Implementing an openflow switch on the netfpga platform. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 1–9. ACM.

NetFPGA (2015a). Netfpga cml. Disponível em: http://netfpga.org/site/#/systems/2netfpga-1g-cml/details/. Acessado em: 27/12/2015.

NetFPGA (2015b). Netfpga sume. Disponível em: http://netfpga.org/site/#/systems/1netfpga-sume/details/. Acessado em: 27/12/2015.

NetFPGA (2015c). Project switch openflow netfpga. Disponível em: https://github.com/NetFPGA/netfpga/wiki/OpenFlowNetFPGA100. Acessado em: 22/11/2015.

ONF (2012). Software-defined networking: The new norm for networks. Disponível em: http://www.opennetworking.org. Acessado em: 23/11/2015.

OpenNetworkingLab (2015). Controller pox. Disponível em: https://openflow.stanford.edu/display/ONL/POX+Wiki. Acessado em: 27/12/2015.

Prasad, R., Dovrolis, C., Murray, M., and Claffy, K. (2003). Bandwidth estimation: metrics, measurement techniques, and tools. *Network, IEEE*, 17(6):27–35.

Rasley, J., Stephens, B., Dixon, C., Rozner, E., Felter, W., Agarwal, K., Carter, J., and Fonseca, R. (2014). Planck: millisecond-scale monitoring and control for commodity networks. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 407–418. ACM.

Van Adrichem, N. L., Doerr, C., Kuipers, F., et al. (2014). Opennetmon: Network monitoring in openflow software-defined networks. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–8. IEEE.

Wellem, T., Lai, Y.-K., and Chung, W.-Y. (2015). A software defined sketch system for traffic monitoring. In *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems*, pages 197–198. IEEE Computer Society.

Yassine, A., Rahimi, H., and Shirmohammadi, S. (2015). Software defined network traffic measurement: Current trends and challenges. *Instrumentation & Measurement Magazine, IEEE*, 18(2):42–50.

Yu, M., Jose, L., and Miao, R. (2013). Software defined traffic measurement with opensketch. In *NSDI*, volume 13, pages 29–42.

Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S., Yahia, M. H., and Zhang, M. (2015). Congestion control for large-scale rdma deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 523–536. ACM.

Coleta e Análise de Características de Fluxo para Classificação de Tráfego em Redes Definidas por Software

Rodolfo Vebber Bisol[1], **Anderson Santos da Silva**[1], **Cristian Cleder Machado**[1], **Lisandro Zambenedetti Granville**[1], **Alberto Egon Schaeffer-Filho**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

Email: {rvbisol, assilva, ccmachado, granville, alberto}@inf.ufrgs.br

Resumo. *Uma classificação acurada de fluxos de tráfego é uma tarefa muito importante no âmbito de resiliência de redes de computadores, podendo ser aplicada para diversos objetivos, por exemplo, criar mecanismos para detecção de fluxos de tráfego maliciosos. Estes mecanismos de classificação de fluxos de tráfego, apesar de já serem muito sofisticados, ainda precisam ser aprimorados, pois certos ataques podem se camuflar entre os tráfegos benignos, que normalmente existem na rede, o que pode dificultar a sua classificação. O objetivo deste trabalho é aprimorar as técnicas de seleção de características e classificação de tráfego propostas em trabalhos anteriores pela adição dos algoritmos Sequential Backward Selection (SBS) e K-means. Além disso, como validação dessa extensão, este trabalho realiza experimentos em um ambiente mais realístico, onde diversos tipos de tráfego coexistem e camuflam-se na rede. Os resultados dos experimentos indicam que a combinação do algoritmo SBS com os algoritmos de classificação de tráfego SVM e K-means permite a identificação de características adequadas para identificar tráfego malicioso.*

Abstract. *An accurate flow classification is a very important task in network resilience and can be applied with many purposes such as to create mechanisms for malicious traffic detection. These mechanisms although sophisticated still need to be refined because certain attacks can hide amongst benign flows that normally exist in the network which can hinder their classification. The main objective of this work is to enhance the feature selection and flow classifier techniques proposed in previous work by the addition of the Sequential Backward Selection (SBS) and K-means algorithms. Moreover, as validation of our extension, this work conducts the experimental evaluation on a more realistic environment, where different flow profiles coexist and camouflage in the network. The experiment results indicate that the combination of SBS with the SVM and K-means traffic classification algorithms allows the identification of appropriate features to identify malicious traffic.*

1. Introdução

Redes de computadores e principalmente a Internet se tornaram essenciais para a vida moderna, sendo utilizadas por governos, empresas e instituições de ensino. Por isso, é muito importante identificar comportamentos anômalos e ataques de usuários maliciosos a fim de manter o adequado funcionamento das redes.

A classificação de fluxos de tráfego pode ser utilizada para diversos objetivos, como detecção de ataques, realocação de recursos de redes e modelagem de perfil de usuários [Nguyen 2008]. Para se obter uma classificação acurada, é necessário identificar diversas características, tais como duração do fluxo e número de pacotes transmitidos. Essas características são capazes de descrever fielmente os diferentes perfis de tráfego que possam coexistir na rede, por exemplo, o número de pacotes e bytes transmitidos em fluxos de tráfego HTTP é caracterizado por rajadas, diferentemente de fluxos de tráfego FTP que apresentam-se de maneira contínua. A operação de coleta de características é dificultada por limitações existentes na organização de redes IP tradicionais, tais como: *(i)* o acesso à informação sobre o estado da rede é heterogêneo para dispositivos de diferentes fabricantes; *(ii)* a grande complexidade da adição de novas funcionalidades aos *switches* ou roteadores de rede, já que cada equipamento deve ser modificado individualmente; e *(iii)* a lógica distribuída em cada elemento de rede pode alterar um fluxo durante seu trânsito pela rede, por exemplo, fazendo alterações na prioridade dos pacotes ou em cabeçalhos TCP/IP ou IP.

A adoção de Redes Definidas por Software (Software Defined Networking - SDN), neste contexto, elimina as limitações mencionadas e facilita o estudo e implementação de novas técnicas que tornam a classificação de fluxos de tráfego mais acurada [ONF 2012]. O fato de SDN ser caracterizada pela separação dos planos de controle e encaminhamento faz com que a lógica esteja centralizada em controladores SDN, como o NOX [Gud 2008] e FloodLight [Erickson 2012]. Assim, alterações na lógica da rede são realizadas apenas nos controladores. Além disso, a coleta de dados sobre os fluxos de tráfego existentes na rede também se torna uma tarefa menos complexa, sendo realizada através dos controladores que possuem visão de grandes porções da rede, ou até mesmo global quando utilizados em conjunto [Machado et al. 2014]

Mesmo SDN tornando esta tarefa de coleta de informações mais simples e acurada, a adoção de SDN por si só não torna os mecanismos de classificação de fluxos de tráfego mais acurados. Outras medidas devem ser tomadas para aumentar a precisão destes mecanismos, tais como uma análise detalhada das características criadas através do processo de seleção de características. Esse processo consiste em identificar quais características são mais relevantes para a classificação de fluxos de tráfego e eliminar características correlacionadas que podem adicionar ruídos à classificação ou gerar desperdício de recursos computacionais [Kim et al. 2008].

Dentro deste cenário, o objetivo deste artigo é aprimorar o trabalho desenvolvido anteriormente em [da Silva et al. 2015] que, através de contadores nativos tais como número de pacotes e bytes, estendeu um conjunto de características (*e.g.*, tamanho médio dos pacotes e tempo entre chegada dos pacotes). Este artigo apresenta a implementação de um novo algoritmo de seleção de características, o *Sequential Backward Selection* (SBS), que é analisado e comparado com estratégias baseadas em algoritmo genético e Análise de Componente Principal (*Principal Component Analysis* - PCA). Além disso, é proposta a utilização de técnicas de seleção de características para aprimorar a precisão de dois algoritmos de classificação de fluxos de tráfego, SVM e K-means. Como estudo de caso e validação da extenção proposta, foram realizados experimentos utilizando transferências FTP e *streaming* de vídeo como tráfegos benignos e ataques DDoS como tráfegos maliciosos.

Este artigo está estruturado na seguinte forma. Na Seção 2, é apresentada uma visão geral dos assuntos abordados neste trabalho, incluindo SDN, seleção de características e classificação de fluxos de tráfego. Na Seção 3, é descrito o sistema de coleta e extensão de características de fluxo de tráfego e os algoritmos utilizados. Na Seção 4, são apresentados os detalhes sobre implementação, incluindo a parametrização dos algoritmos, ferramentas utilizadas e resultados obtidos. Finalmente, na Seção 5, este trabalho é concluído apresentando as considerações finais e propostas para trabalhos futuros.

2. Contextualização e Trabalhos Relacionados

Um classificador de fluxos de tráfego é um processo que captura diferentes fluxos de tráfego e determina a qual classe de tráfego tais fluxos pertencem. Essa ação é realizada através da análise dos dados dos pacotes, (*e.g.*, os endereços IP de origem e de destino) [Blake et al. 1998]. A classificação de fluxos de tráfego é utilizada para identificar os perfis de tráfego existentes na rede. A classificação dos fluxos de tráfego é importante para auxiliar a administração de redes de computadores, sendo utilizada para monitoramento, filtro de conteúdo, avaliação de qualidade de serviço (QoS), entre outras aplicações.

A utilização de um grande conjunto de características para a classificação de fluxos de tráfego possui desvantagens, tais como *(i)* o uso de características irrelevantes pode adicionar ruído, dificultando a classificação, e *(ii)* o processamento de um grande conjunto de características, possivelmente correlacionadas, pode gerar desperdício de recursos computacionais. Procurando resolver este problema, foram desenvolvidas diversas técnicas de seleção de características [Blum and Langley 1997]. Tais técnicas consistem em determinar as características, ou atributos, mais relevantes para um determinado problema e eliminar as características menos relevantes. A seleção de características oferece uma série de vantagens, tais como *(i)* um melhor entendimento e visualização dos dados, *(ii)* redução do tempo de treinamento de algoritmos de aprendizado de máquina (*Machine Learning* - ML), e *(iii)* uma maior acurácia na classificação de fluxos [Guyon 2003].

Por não ser uma tarefa trivial, existem hoje diversas técnicas de seleção de características, que podem ser divididas em três grupos principais [Sae 2007]. O primeiro grupo é o das técnicas conhecidas como filtros, que analisam as propriedades intrínsecas dos dados através de métodos matemáticos, por exemplo matrizes de correlação. Além disso, tais técnicas são independentes de algoritmos de classificação ou indução. Exemplos de filtros são Análise de Componente Principal (*Principal Component Analysis* - PCA) [Jolliffe 1986] e *Fast Correlation-based Filter* (FCBF) [Yu 2003]. O segundo grupo de técnicas é conhecido como *Wrappers*. Essas técnicas possuem como característica o uso de métodos de indução como uma sub-rotina e realizam uma busca heurística dentro do espaço de possíveis subconjuntos. Exemplos de *Wrappers* são o OBLIVION [Langley 1994] e os Algoritmos Genéticos Híbridos [Oh et al. 2004]. Por último, existem as técnicas *Embedded*. Essas técnicas são desenvolvidas especificamente para um determinado problema e possuem um funcionamento parecido com o dos *Wrappers*.

O surgimento do paradigma SDN trouxe novas possibilidades que podem solucionar problemas em relação à coleta e extensão de características de fluxos de tráfego utilizadas para a classificação dos mesmos [Feamster et al. 2013]. Isto se deve por dois motivos: o primeiro é a separação dos planos de controle e encaminhamento e, consequentemente, a centralização da lógica da rede em um controlador. Desta forma, a coleta

de informações sobre os fluxos existentes na rede é facilitada, passando a ser feita através do controlador da rede. E segundo, o protocolo OpenFlow [McKeown et al. 2008], que define a comunicação entre o controlador da rede e os *switches*, que por sua vez possuem em sua especificação uma tabela de contadores referentes aos fluxos de tráfego existentes, por exemplo contadores de *bytes* e pacotes. Esses contadores foram definidos com o objetivo de facilitar a criação de mecanismos de QoS, por exemplo mecanismos de classificação de tráfego [ONF 2014].

3. Arquitetura para Coleta e Análise de Características em SDN

Esta seção descreve uma visão geral do trabalho realizado anteriormente em [da Silva et al. 2015], indicando como é realizado o processo de coleta e extensão de características de fluxos de tráfego. Além disso, apresenta os algoritmos de seleção de características e classificação de tráfego implementados inicialmente, *i.e.*, Análise do Componente Principal (*Principal Component Analysis* - PCA), Algoritmo Genético (*Genetic Algorithm* - GA) e Máquina de Vetores de Suporte (Support Vector Machine - SVM). Finalmente, são descritos os algoritmos propostos para extensão da arquitetura. Esses algoritmos são o Seleção Sequencial Inversa (*Sequential Backward Selection* - SBS) e o K-means.

3.1. Arquitetura

A arquitetura da solução é representada na Figura 1. A arquitetura consiste em dois componentes principais: *(i) Flow Feature Manager*, responsável pela coleta de informações da rede e criação de um conjunto de características avançadas que possam representar fielmente cada fluxo existente; e *(ii) Flow Feature Selector*, responsável por definir o melhor subconjunto de características para classificação de fluxos de tráfego. A seguir, os dois componentes e seus módulos são descritos em maiores detalhes:

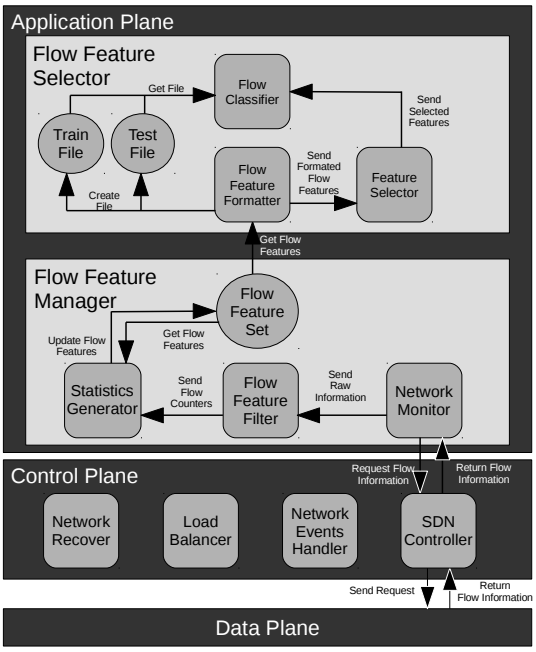


Figura 1. Arquitetura para coleta e extensão de características de fluxo.

- ***Flow Feature Manager –*** inclui os módulos: *Network Monitor Module*, responsável por coletar informações da rede; *Flow Feature Filter Module*, responsável por filtrar as informações recebidas selecionando somente as informações relevantes ao sistema; e *Statistics Generator Module*, responsável por criar o conjunto de características avançadas com as informações recebidas do *Flow Feature Filter Module*. O conjunto de características criado é representado pela estrutura de dados *Flow Feature Set*.
- ***Flow Feature Selector –*** inclui os módulos: *Flow Feature Formatter Module*, responsável por reorganizar o conjunto de características em um formato padrão e criar os arquivos de treinamento e teste; *Feature Selector Module*, responsável por selecionar as características de fluxo mais importantes e criar subconjuntos formados por estas características; e *Flow Classifier Module*, responsável por avaliar a qualidade dos subconjuntos criados através da classificação de fluxos de tráfego.

3.2. Coleta e Extensão de Características de Fluxo

Inicialmente, o *Network Monitor*[1] envia uma requisição ao controlador solicitando informações sobre os fluxos existentes na rede. O controlador, por sua vez, reúne as informações presentes nas tabelas de encaminhamento dos *switches* e as envia para o *Network Monitor*. Este, repassa as informações para o *Flow Feature Filter* que seleciona apenas as informações que serão utilizadas, por exemplo: valor dos contadores de *bytes* e pacotes que são utilizados para a criação do conjunto de características. Além desses contadores, o *Flow Feature Filter* também seleciona os endereços IP e portas TCP/UDP de origem e destino e o protocolo da camada de transporte, de modo que cada fluxo seja identificado univocamente. O *Statistics Generator* recebe os identificadores de fluxo e os dois contadores e estende estes contadores em características de fluxo mais representativas, como tamanho dos pacotes e duração do fluxo. O *Statistics Generator* também é responsável por criar e manter atualizada uma estrutura de dados, o *Flow Feature Set*, que reúne todas as características de cada fluxo de tráfego identificado na rede.

As informações utilizadas para a criação das características de fluxos de tráfego são dois contadores nativos, presentes na tabela de contadores do protocolo OpenFlow: *Byte Counter* e *Packet Counter*. Para estender os contadores nativos do OpenFlow, o sistema utiliza outros dois valores criados internamente: contador de amostras, que salva a quantidade de vezes que um determinado fluxo foi amostrado na rede, e tempo entre requisições enviadas ao controlador pelo *Network Monitor*.

A partir destas informações, são criadas quatro características básicas: *bytes* por segundo; pacotes por segundo; comprimento dos pacotes; e intervalo entre a chegada de dois pacotes. Estas quatro características são estimativas dos valores reais, uma vez que para se obter os valores reais de cada uma delas seria necessário coletar informações em nível de pacote, o que não é oferecido nativamente pelo OpenFlow e demandaria uma maior capacidade computacional por parte da aplicação. Estas características isoladamente não oferecem informações suficientes sobre o perfil ou o estado dos fluxos de tráfego. Com o objetivo de obter um maior entendimento sobre o comportamento dos tráfegos, estas quatro características são expandidas em três grupos de características: *Características Escalares*, *Características Estatísticas* e *Características Complexas*.

[1]Utiliza-se uma API REST para obter dados do controlador a cada 30s (intervalo configurável em nossa implementação).

As *Características Escalares* representam o estado dos fluxos de tráfego em um determinado tempo. As *Características Escalares* são: máximo e mínimo de cada uma das quatro características básicas, o tamanho dos fluxos em *bytes* e o número de pacotes, e a duração dos fluxos. As *Características Estatísticas* oferecem um entendimento mais preciso sobre o comportamento dos fluxos de tráfego ao longo do tempo e são elas: as médias e variâncias das quatro características básicas e o primeiro e terceiro quartis do intervalo de chegada entre dois pacotes, e tamanho dos pacotes. As *Características Complexas* são os dez primeiros componentes da Transformada Discreta de Fourier. A Transformada de Fourier representa o comportamento da função em um domínio de frequência, onde cada componente é um par complexo em que o módulo representa o quanto a frequência (ou amostra) é comum, enquanto o valor complexo representa a fase da frequência. Os dez primeiros componentes da Transformada de Fourier também são avaliadas como características relevantes para a classificação de fluxos de tráfego no trabalho de [Auld et al. 2007]. A Tabela 1 apresenta o conjunto final de trinta e três características de fluxo criadas usando os contadores nativos do OpenFlow.

Tabela 1. Características estendidas utilizando contadores nativos do OpenFlow.

Características Escalares	Características Estatísticas	Características Complexas
Bytes por Segundo Máximo	Média de *Bytes* por Segundo	1^o Componente da Transformada Discreta de Fourier
Bytes por Segundo Mínimo	Variância de *Bytes* por Segundo	2^o Componente da Transformada Discreta de Fourier
Pacotes por Segundo Máximo	Média de Pacotes por Segundo	3^o Componente da Transformada Discreta de Fourier
Pacotes por Segundo Mínimo	Variância de Pacotes por Segundo	4^o Componente da Transformada Discreta de Fourier
Tamanho Máximo de Pacote	Tamanho Médio dos Pacotes	5^oComponente da Transformada Discreta de Fourier
Tamanho Mínimo de Pacote	Variância do Tamanho dos Pacotes	6^o Componente da Transformada Discreta de Fourier
Tempo Máximo Entre a Chegada de Dois Pacotes	Tempo Médio Entre a Chegada de Dois Pacotes	7^o Componente da Transformada Discreta de Fourier
Tempo Mínimo Entre a Chegada de Dois Pacotes	Variância do Tempo Entre a Chegada de Dois Pacotes	8^o Componente da Transformada Discreta de Fourier
Tamnaho Total em *Bytes*	1^o Quartil do Tamanho dos Pacotes	9^o Componente da Transformada Discreta de Fourier
Tamanho Total em Pacotes	3^o Quartil do Tamanho dos Pacotes	10^o Componente da Transformada Discreta de Fourier
Duração do Fluxo	1^o Quartil do Tempo Entre Chegada de Dois Pacotes	
	3^o Quartil do Tempo Entre Chegada de Dois Pacotes	

3.3. Algoritmos de Seleção de Características e Classificação de Tráfego

A seguir são descritos em maiores detalhes os algoritmos de seleção de características e classificação de fluxos de tráfego.

O primeiro algoritmo implementado para a seleção de características é chamado de Análise do Componente Principal (*Principal Component Analysis* - PCA). O método matemático PCA foi desenvolvido por Karl Pearson em 1901 [Pearson 1901]. Esse método avalia a correlação existente entre as diferentes variáveis de um problema, neste caso,

as características dos fluxos de tráfego, criando os chamados Componentes Principais. A quantidade de componentes criadas é igual ao número de características existentes no problema (no caso deste trabalho, trinta e três características). As componentes principais são compostas pelas variáveis originais multiplicadas por um peso. Esse peso varia de -1 a 1 e quanto mais próximo de 1 mais importante a variável é para a componente.

O segundo algoritmo é o Algoritmo Genético. Esse algoritmo realiza uma busca heurística no espaço de solução através da combinação de soluções (*crossover*) e inserção de novas características às soluções (*mutation*), realizando assim um processo de evolução das soluções a cada geração [Haupt 1998]. Os elementos que compõem esta classe de algoritmos são: uma população na qual cada indivíduo é uma possível solução para o problema; uma função de avaliação, sendo que esta função avalia a qualidade de cada indivíduo, ou seja, a qualidade da solução; e a quantidade de gerações que devem ser executadas.

Para a classificação de fluxos de tráfego, foi implementado o algoritmo Máquina de Vetores de Suporte (*Support Vector Machine* - SVM). O algoritmo SVM é utilizado em diversas áreas, desde o reconhecimento de imagens à bioinformática. No contexto de rede de computadores, especificamente para a classificação de tráfego, já foi comprovada a eficácia deste algoritmo para a detecção de ataques maliciosos, tais como DDoS [Mukkamala 2003]. O princípio de funcionamento do SVM é a construção de um hiperplano que separa as classes de um problema em um espaço N-dimensional, onde N é a quantidade de características do problema. Essa separação procura maximizar a distância entre cada uma das classes e é posicionada entre os dois pontos mais próximos pertencentes a duas classes diferentes [Bennett 2000].

3.4. Estendendo os Módulos de Seleção e Classificação

A seguir são descritos os algoritmos propostos para estender a arquitetura descrita em 3.1. Para a seleção de características é proposto o SBS e para a classificação de tráfego é proposto o K-means.

O algoritmo *Sequential Backward Selection* (SBS) é classificado como um filtro. Referente ao seu funcionamento, esse algoritmo elimina as características que mais interferem na classificação. O processo de escolha da característica a ser eliminada consiste em, partindo de um subconjunto, avaliar a qualidade deste subconjunto com cada característica eliminada uma a uma. O subconjunto escolhido para a próxima iteração é o subconjunto inicial com uma característica a menos que apresente o melhor desempenho. O SBS encerra quando, em alguma iteração, todos os possíveis novos subconjuntos resultam em um decréscimo na qualidade.

O SBS não garante que o subconjunto final é o melhor subconjunto possível. O subconjunto final é sempre um máximo local. Isso se deve ao fato de que uma característica, depois de removida, nunca é adicionada novamente ao subconjunto. Dessa forma, o SBS não avalia características que possam se tornar relevantes ao problema com a eliminação de outras características.

O segundo algoritmo proposto, para a classificação de fluxos de tráfego, é o K-means. Esse algoritmo é baseado em aprendizado não supervisionado, e sua utilização é voltada para processos de clusterização. O K-means separa as amostras em um espaço N-dimensional em *clusters*, cuja partição minimiza o erro quadrático médio entre o centro

dos *clusters* e cada ponto pertencente a este. O procedimento, de acordo com MacQueen [MacQueen 1967], consiste em iniciar k *clusters*, cada um sendo um ponto randomicamente selecionado no espaço, e a cada iteração adicionar a cada *cluster* o ponto que minimiza a média da distância entre eles.

4. Experimentos e Resultados

Nesta seção é descrito o ambiente em que o sistema apresentado anteriormente foi implementado, os experimentos realizados, e a parametrização dos algoritmos de seleção de características SBS, PCA e GA. Por fim, são apresentados os resultados obtidos com a classificação de fluxos de tráfego utilizando os algoritmos SVM e K-means.

4.1. Ambiente e Casos de Teste

Para validação do sistema proposto, foi criado um ambiente para testes composto por uma topologia em árvore com vinte e um switches, sessenta e quatro hosts, utilizando o emulador Mininet e um controlador Floodlight. Foi escolhida uma topologia em árvore por ser escalável e muito utilizada na prática. Adicionalmente, sobre esse ambiente são executados um módulo gerador de tráfego e o mecanismo de coleta, extensão e análise de características de fluxos descritos na Seção 3.

Referente à implementação, tanto o sistema quanto os programas geradores de tráfego foram implementados em Python v2.7.6. A estrutura de dados que armazena o conjunto de características é um dicionário cuja chave é a 5-tupla identificadora de fluxo descrita na Seção 3.2, e os arquivos de treinamento e teste são salvos em formato .csv. O algoritmo SBS foi implementado em Python v2.7.6 enquanto os algoritmos SVM, PCA, GA, e K-means foram implementados em R v3.0.2 utilizando as bibliotecas e1071, psych, genalg e cluster.

Os fluxos de tráfego gerados na rede são de quatro tipos diferentes: *(i)* um servidor HTTP é alvo de ataques DDoS partindo de determinados *hosts*; *(ii)* um servidor de stream de vídeo hospedado em um *host* e os demais *hosts* assinam este serviço, sendo que este tráfego é gerado utilizando o programa VLC[2] tanto no servidor quanto nos clientes; *(iii)* trocas de arquivos entre os *hosts* utilizando o protocolo FTP; e *(iv)* tráfego de *background* gerado através do Scapy[3].

Os três cenários de teste criados para a avaliação deste trabalho são compostos pelos quatro perfis de tráfego. A diferença entre cada cenário é a proporção estabelecida para cada perfil de tráfego. Tais proporções foram definidas seguindo a quantidade de fluxos que é normalmente encontrada na rede [Isolani et al. 2015]. A Tabela 2 apresenta os três cenários e a proporção que cada perfil de tráfego representa em cada cenário. Para os cenários B e C é aumentada a porcentagem de ataques DDoS e é diminuída a porcentagem dos demais perfis de tráfego. Assim, é possível observar o impacto que a proporção de fluxos utilizada no treinamento possui na acurácia do algoritmo de classificação de fluxos de tráfego.

Em cada cenário de teste são utilizados no total quinze mil amostras de treinamento e duas mil e quinhentas amostras de teste para o SVM. Igualmente, são utilizadas quinze mil amostras para a clusterização realizada pelo K-means.

[2]http://www.videolan.org/vlc/index.html

[3]http://www.secdev.org/projects/scapy/

Tabela 2. Proporção de volume de tráfego para cada cenário de teste.

	Ataques DDoS	Tráfego FTP	Stream de Vídeo	Background
Cenário A	10%	10%	50%	30%
Cenário B	30%	10%	40%	20%
Cenário C	50%	7.5%	25%	17.5%

Para permitir a comparação entre a qualidade de cada subconjunto criado, foram utilizadas as métricas de acurácia para o algoritmo SVM, ou seja, a porcentagem de amostras de ataques DDoS identificadas, e o método de Silhouettes [Rousseeuw 1987] para o algoritmo K-means. Os valores de acurácia variam de 0% a 100% e Silhouettes de -1 a 1, sendo 1 a melhor qualidade.

4.2. Parametrização de Algoritmos

O SBS só possui um parâmetro a ser definido: a quantidade de características a serem eliminadas. O valor definido é o tamanho inicial do conjunto de características, *i.e*, trinta e três, pois quanto menor o conjunto de características menor a demanda de recursos para a classificação dos fluxos de tráfego.

O PCA e o GA possuem seus parâmetros definidos de acordo com o trabalho realizado por [da Silva et al. 2015]: são gerados trinta e três componentes principais utilizando os pesos 0.025, 0.05 e 0.1 como *threshold* para o PCA; e para o GA são executadas cem gerações com uma população de duzentos indivíduos e probabilidade de mutação 0.01.

4.3. Experimentos e Resultados da Classificação de Fluxos de Tráfego

O módulo *Flow Classifier* foi instanciado com os algoritmos SVM e K-means. Esses algoritmos foram escolhidos por utilizarem duas formas diferentes de ML: enquanto o SVM utiliza aprendizado supervisionado, o K-means realiza um processo de clusterização não supervisionado. As subseções seguintes apresentam os resultados dos experimentos realizados para os dois algoritmos de classificação.

4.3.1. SVM

Inicialmente, é avaliada a classificação de fluxos utilizando o SVM. Para tanto, foi comparada a acurácia da classificação utilizando o conjunto completo, com a acurácia utilizando os subconjuntos criados por cada um dos três algoritmos de seleção de características. Os gráficos da Figura 2 apresentam a evolução da acurácia do subconjunto a cada iteração do algoritmo SBS. Podem ser observadas nesses gráficos duas informações importantes: primeiro, a última iteração resulta em um decréscimo da acurácia. Isto indica o momento em que o algoritmo interrompe a execução. Segundo, existem momentos em que a acurácia não sofre alteração (cenários A e B). Isto mostra que o algoritmo sempre busca o menor subconjunto possível, mesmo quando a acurácia não se altera com a eliminação de uma característica.

A Tabela 3 compara a acurácia obtida com o melhor subconjunto de características

criado por cada algoritmo de seleção de características[4]. O tamanho de cada subconjunto de características é informado entre parênteses após a acurácia do mesmo.

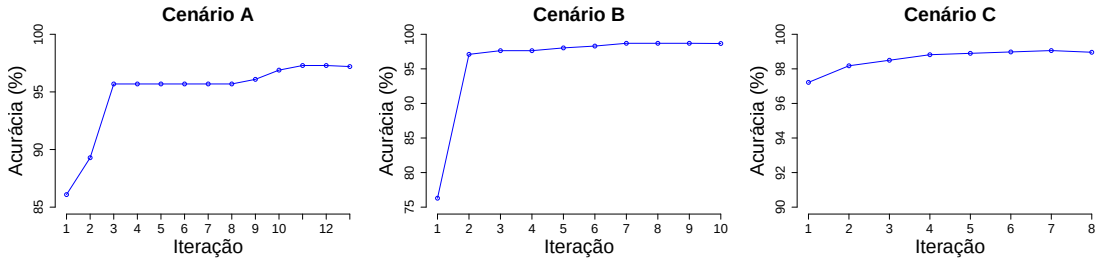


Figura 2. Evolução da acurácia do subconjunto a cada iteração do algoritmo SBS.

Tabela 3. Comparativo entre a acurácia dos subconjuntos criados para a classificação utilizando algoritmo SVM.

	Todas Características	SBS	PCA	Algoritmo Genético
Cenário A	86%	97.2% (20)	94.8% (12)	97.2% (21)
Cenário B	76.26%	98.53% (24)	96.93% (8)	98.67% (21)
Cenário C	97.2%	99.04% (26)	99.04% (10)	99.12% (21)

4.3.2. K-means

O K-means possui alguns parâmetros a serem definidos, incluindo: a quantidade de *clusters* a serem criados, e a quantidade mínima e máxima de iterações. Devido ao fato de cada um dos três cenários de teste possuir quatro perfis de tráfego diferentes, o K-means é configurado para criar quatro *clusters*, cada um representando um perfil de tráfego diferente. Além disso, para as quantidades mínimas e máximas de iterações são utilizados os valores padrão de cem e mil iterações, respectivamente.

Igualmente ao SVM, foram executados os três algoritmos de seleção de características com o objetivo de maximizar a Silhoutte da clusterização realizada pelo K-means. Os parâmetros dos algoritmos de seleção de características utilizados foram os mesmos utilizados para o SVM. A Figura 3 apresenta a evolução da Silhouette a cada iteração do algoritmo SBS. Adicionalmente, a Tabela 4 apresenta os resultados para cada um dos três algoritmos de seleção de características e o tamanho de cada subconjunto.

As figuras 4, 5 e 6 apresentam a clusterização realizada pelo K-means para os cenários A, B e C, respectivamente. Como pode ser observado, para cada figura, são comparados os resultados do conjunto de características completo (à esquerda) com os os resultados dos subconjuntos criados por cada um dos algoritmos de seleção de características.

[4]Os resultados obtidos para os algoritmos de seleção PCA e Algoritmo Genético não são detalhados nesse artigo por restrições de espaço. No entanto, informações adicionais podem ser encontradas em [da Silva et al. 2015].

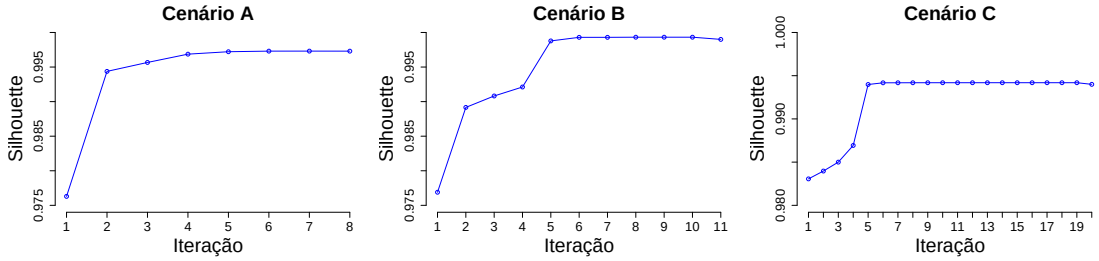


Figura 3. Evolução da qualidade da clusterização do subconjunto a cada iteração do SBS.

Tabela 4. Tabela comparativa entre a Silhueta dos subconjuntos criados para clusterização utilizando K-means.

	Todas Características	SBS	PCA	Algoritmo Genético
Cenário A	0.976295	0.997305 (25)	0.997305 (12)	0.983541 (24)
Cenário B	0.976893	0.999317 (23)	0.999092 (5)	0.991594 (17)
Cenário C	0.983061	0.994187 (14)	0.988336 (15)	0.983100 (24)

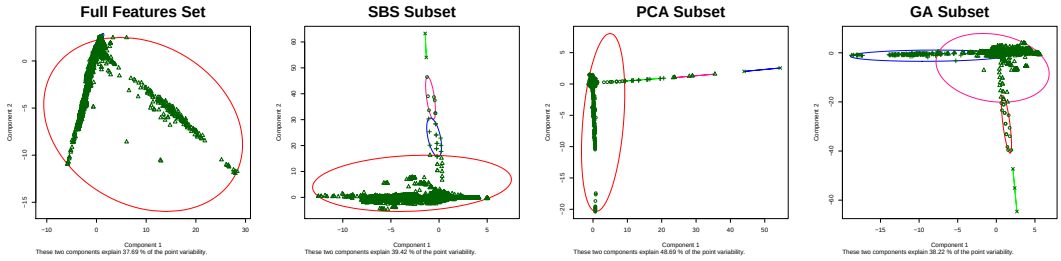


Figura 4. Resultado da clusterização para o cenário A.

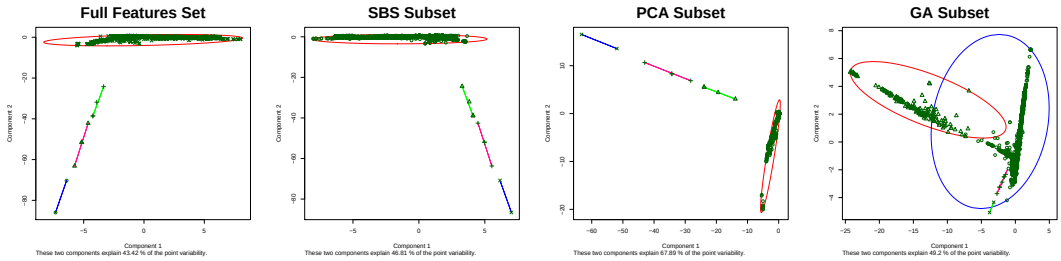


Figura 5. Resultado da clusterização para o cenário B.

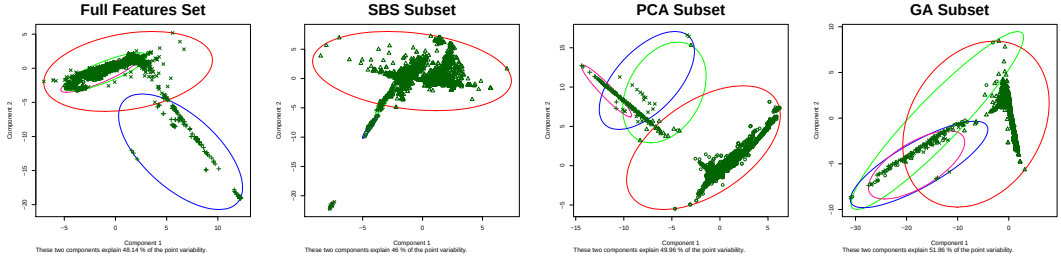


Figura 6. Resultado da clusterização para o cenário C.

4.3.3. Discussão sobre os resultados obtidos

Os gráficos apresentados na Figura 7 resumem os resultados obtidos com a execução dos algoritmos de classificação utilizando o conjunto completo de características e os subconjuntos compostos pelas características selecionadas pelos algoritmos de seleção. Em complemento, a Tabela 5 apresenta o tempo de execução de cada algoritmo de seleção de características.

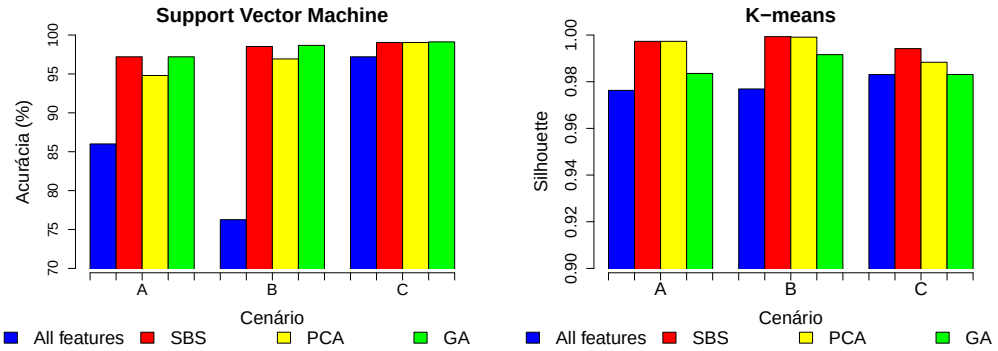


Figura 7. Gráficos comparativos da qualidade da solução oferecida pela utilização dos três algoritmos de seleção de características.

Tabela 5. Tempo de execução dos algoritmos de seleção de características.

	SVM			K-Means		
	SBS	**PCA**	**GA**	**SBS**	**PCA**	**GA**
Cenário A	21min 42s	1min 51s	16h 36min	22min 10s	3min 13s	35h 42min
Cenário B	19min 34s	2min 07s	15h 40min	26min 14s	2min 59s	29h 42min
Cenário C	12min 38s	1min 47s	11h 11min	42min 57s	3min 06s	35h 59min

Observando o gráfico à esquerda na Figura 7, pode ser observado que em todos os três cenários existe uma evolução na acurácia do algoritmo SVM, mostrando que a seleção de caraterísticas tem um impacto positivo na classificação de fluxos de tráfego utilizando o SVM. Além disso, o SBS apresenta um melhor desempenho que os outros dois algoritmos por apresentar uma acurácia parecida a do algoritmo genético e um tempo de execução de em média 17min58s.

Semelhantemente, observando os valores de Silhouette para cada um dos subconjuntos, pode-se afirmar que existe uma melhoria na qualidade da clusterização. Assim, identificando a qual classe uma das amostras de um *cluster* pertence, é possível afirmar com mais acurácia que todas as demais amostras do *cluster* pertencem à mesma classe. Além disso, observando o tempo de execução e o resultado obtido por cada algoritmo de seleção de características, é possível afirmar que o SBS apresenta um desempenho superior aos demais, por apresentar o subconjunto com a melhor Silhouette e um tempo de execução de em média 30min27s.

5. Conclusão

Neste trabalho, foi explorada uma arquitetura modular estendendo seus módulos de seleção de características e classificação de tráfego com mais dois algoritmos: Seleção

Sequencial Inversa (Sequential Backward Selection - SBS) e K-means. Os cenários e perfis de tráfego utilizados nos experimentos demonstram que a precisão dos algoritmos de classificação depende tanto do conjunto utilizado para treinamento e clusterização de algoritmos de aprendizado supervisionado (SVM) quanto da seleção de características. Através dos resultados apresentados nos experimentos, pode ser observado que é possível alcançar níveis de precisão maiores aumentando a proporção de ataques DDoS utilizados para o treinamento do SVM. Baseando-se nos resultados obtidos, também é possível concluir que o algoritmo SBS tem um desempenho muito bom por apresentar uma grande melhoria na qualidade da classificação do SVM e na clusterização do K-means e também um tempo de execução muito inferior ao algoritmo genético sem perda de qualidade.

Possíveis extensões que podem ser adicionadas a este trabalho incluem a utilização de tráfego real e não apenas tráfego sinteticamente criado. Além disso, pretende-se analisar outras abordagens para a coleta de características como *Deep Packet Inspection* (DPI) e também investigar a criação de novas características de fluxos de tráfego.

Agradecimentos

Este trabalho é apoiado por ProSeG - Segurança da Informação, Proteção e Resiliência em Smart Grids, um projeto de pesquisa financiado pelo MCTI/CNPq/CT-ENERG # 33/2013.

Referências

(2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517.

(2008). NOX: Towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110.

Auld, T., Moore, A., and Gull, S. (2007). Bayesian neural networks for internet traffic classification. *Neural Networks, IEEE Transactions on*, 18(1):223–239.

Bennett, Kristin P. e Campbell, C. (2000). Support vector machines: Hype or hallelujah? *SIGKDD Explor. Newsl.*, 2(2):1–13.

Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. (1998). RFC2475 - An Architecture for Differentiated Services. RFC 2475, RFC Editor.

Blum, A. L. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271.

da Silva, A. S., Machado, C. C., Bisol, R. V., Granville, L. Z., and Schaeffer-Filho, A. (2015). Identification and selection of flow features for accurate traffic classification in SDN. *14th IEEE International Symposium on Network Computing and Applications (NCA 2015)*, pages 137–141.

Erickson, D. (2012). Floodlight java based openflow controller. http://www.projectfloodlight.org/floodlight/.

Feamster, N., Rexford, J., and Zegura, E. (2013). The road to SDN. *Queue*, 11(12):20:20–20:40.

Guyon, Isabelle e Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

Haupt, Randy L. e Haupt, S. E. (1998). *Practical Genetic Algorithms*. John Wiley & Sons, Inc., New York, NY, USA.

Isolani, P. H., Araujo Wickboldt, J., Both, C., Rochol, J., and Zambenedetti Granville, L. (2015). Interactive monitoring, visualization, and configuration of openflow-based SDN. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 207–215.

Jolliffe, I. (1986). *Principal Component Analysis*. Springer Verlag.

Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., and Lee, K. (2008). Internet traffic classification demystified: Myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 11:1–11:12, New York, NY, USA. ACM.

Langley, Pat e Sage, S. (1994). Oblivious decision trees and abstract cases. In *Proc. AAAI-94 Workshop on case-based reasoning*, pages 113–117.

Machado, C. C., Granville, L. Z., Schaeffer-Filho, A., and Wickboldt, J. A. (2014). Towards SLA policy refinement for QoS management in software-defined networking. In *Advanced Information Networking and Applications (AINA-2014), 2014 28th IEEE International Conference on*, pages 397–404. IEEE.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Mukkamala, S. e Sung, A. (2003). Detecting denial of service attacks using support vector machines. In *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, volume 2, pages 1231–1236 vol.2.

Nguyen, T.T.T. e Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *Comms. Surveys Tutorials, IEEE*, 10(4):56–76.

Oh, I.-S., Lee, J.-S., and Moon, B.-R. (2004). Hybrid genetic algorithms for feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1424–1437.

ONF (2012). Software defined netwoking: The new norm for networks. https://www.opennetworking.org.

ONF (2014). Openflow switch specification v1.5.0. https://www.opennetworking.org/.

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572.

Rousseeuw, P. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65.

Yu, Lei e Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. pages 856–863.

Avaliação de Desempenho de Planos de Dados OpenFlow

Leonardo C. Costa[1], **Alex B. Vieira**[1]
Erik de Britto e Silva[2], **Daniel F. Macedo**[2]
Geraldo Gomes[3], **Luiz H. A. Correia**[3], **Luiz F. M. Vieira**[2]

[1]Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF) – Juiz de Fora, MG – Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

[3]Departamento de Ciência da Computação
Universidade Federal de Lavras (UFLA) – Lavras, MG – Brasil

leonardocosta@ice.ufjf.br, alex.borges@ufjf.edu.br

{erik,damacedo,lfvieira}@dcc.ufmg.br

gccgomes@sistemas.ufla.br, lcorreia@dcc.ufla.br

Abstract. *Software Defined Networks are characterized by decoupling the data plane and the control plane of network switches. The OpenFlow protocol implements this concept, and due to the advantages of lower operation expenditures and greater innovation in the network, it is found today in various equipment sold by many companies. Despite being widely employed in industry and research, in the literature does not exist a systematic evaluation of OpenFlow data plane performance. In this paper we present a comparison of different OpenFlow implementations, both on hardware and software switches, in order to validate the maturity of existing commercial solutions. We also compare some existing equipment operating in OpenFlow mode versus its traditional mode of operation. Results show strong evidences that some vendors implement OpenFlow in software, and also that the choice of implementation significantly impacts the performance of the switch.*

Resumo. *As Redes Definidas por Software (SDN) são marcadas pela separação dos planos de dados e de controle. O protocolo OpenFlow é uma implementação desse conceito que, devido às vantagens de menor custo de operação e maior inovação na rede, é encontrado hoje em diversos equipamentos comercializados por várias empresas. Apesar de ser muito utilizado na indústria e na pesquisa, não existe na literatura uma avaliação sistemática do desempenho do plano de dados OpenFlow. Neste trabalho apresentamos uma comparação de diferentes implementações OpenFlow, tanto em hardware quanto software, tendo em vista validar a maturidade das soluções comerciais existentes no mercado, e se os equipamentos existentes, quando operando em modo OpenFlow, operam com qualidade igual ou superior ao modo tradicional de operação. Os resultados mostram fortes indícios de que alguns fabricantes implementam OpenFlow em software, e também que a escolha da implementação impacta significativamente no desempenho do switch.*

1. Introdução

O paradigma de Redes Definidas por *Software* (SDN) vem mudando a forma de como criar, modificar e gerenciar as redes de computadores [Feamster et al. 2014]. Ao separar o plano de controle do plano de dados, a evolução e a inovação da rede são facilitadas. Assim, dentre outras coisas, novos serviços podem ser implantados mais rapidamente, diminuindo o custo operacional das redes. Além disso, novas pesquisas se tornaram possíveis em ambientes reais, sem que se afetasse o tráfego ou a disponibilidade da rede de produção [Feamster et al. 2014].

Considerando esse contexto, o paradigma SDN tem atraído a atenção tanto da comunidade acadêmica quanto da indústria. Apesar de existirem propostas mais flexíveis de SDN, tais como o POF e o P4 [Song 2013, Bosshart et al. 2014], o OpenFlow, que está intimamente ligado a SDN, tem recebido uma grande atenção da indústria. Parte dessa atenção deve-se a simplicidade de se implementar OpenFlow nos projetos existentes de comutadores e roteadores. Isso despertou o interesse de vendedores de equipamentos, operadores de redes, e de empresas como Google, Microsoft e Facebook, empresas essas que, em 2011, formaram a Open Networking Foundation (ONF), que tem o objetivo de promover a tecnologia OpenFlow e o paradigma SDN no mercado [ONF 2015]. Como consequência do crescente interesse da indústria pelo OpenFlow, cada vez mais equipamentos de rede, como *switches* e roteadores, passaram a suportar o protocolo [Kreutz et al. 2015]. Mais ainda, diversas redes de produção, como a WAN da Google [Jain et al. 2013], bem como redes de proporções continentais como a rede de pesquisa e cooperação na America Latina AmLight [Ibarra et al. 2015] já utilizam SDN.

Entretanto, uma solução inovadora pode não se tornar um padrão da indústria devido a limitações de desempenho, segurança ou outros fatores. Por exemplo, o paradigma de redes ativas, proposto no início do milênio, também permitia a programação de comutadores [Tennenhouse et al. 1997]. Devido à sua escalabilidade limitada e problemas de segurança, não foi adotada em redes comerciais.

Tendo em vista a popularidade do OpenFlow, é importante verificar se as suas implementações existentes, tanto na forma de equipamentos comerciais quanto em implementações via *software open source*, possuem características de desempenho e robustez adequadas para o uso em redes de produção. Apesar de existirem alguns trabalhos na literatura que avaliam o desempenho dos controladores OpenFlow [Khondoker et al. 2014], ainda são raros os trabalhos que enfocam o plano de dados dos *switches* OpenFlow, bem como suas implementações e mecanismos internos [Bianco et al. 2010, Sünnen 2011, Appelman and De Boer 2012]. Esses trabalhos são esforços no sentido de avaliar o desempenho do OpenFlow em comparação com os modos normais de operação de um *switch* e também em comparar diferentes implementações de *switches* OpenFlow, mas considerando sua relação com o plano de controle. Existe, portanto, uma lacuna na avaliação de desempenho dos *switches* que implementam o protocolo OpenFlow quando consideramos apenas o plano de dados, descartando qualquer interação entre controlador e *switches*.

Tendo em vista esse contexto, o presente trabalho tem como objetivo avaliar a qualidade de diferentes implementações OpenFlow de *hardware switches* comerciais, além de implementações *open source* de *software switches*. Serão avaliadas métricas tradicionais de desempenho, que levam em conta aspectos como latência e jitter, em de-

terminadas operações do *switch*. Assim, com este trabalho, verificaremos se um administrador de rede poderia comprar um *switch* compatível com OpenFlow com a mesma facilidade que hoje se compra um *switch* Ethernet, ou se a compatibilidade e desempenho das implementações ainda são altamente dependentes do fabricante e modelo do equipamento. Deve-se ressaltar que, nessa comparação, procuramos analisar apenas aspectos referentes ao plano de dados dos *switches*, como encaminhamento de pacotes ou outros mecanismos internos, descartando a influência do controlador nas operações analisadas. Além disso, fizemos comparações entre o desempenho de cada *switch* em seus modos *legacy* (normal) e OpenFlow, para identificar se há vantagem no uso do protocolo OpenFlow. Finalmente, procuramos verificar se é possível identificar, a partir de métricas de desempenho da rede, se as implementações do protocolo são realizadas, de fato, em *hardware*, ou se são apenas instâncias de um *software switch* adaptadas ao mecanismo interno do *switch*. Os resultados mostram que ainda existem diferenças significativas de desempenho e funcionalidade nas implementações em *hardware*.

O restante deste artigo está organizado da seguinte forma. A seção 2 apresenta os trabalhos relacionados. A seção 3 descreve a metodologia de avaliação, enquanto a seção 4 discute os resultados obtidos. A seção 5 conclui o artigo, apresentando as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O advento do paradigma de Redes Definidas por Software (SDN) vem mudando a forma como projetamos e gerenciamos as redes de computadores. Segundo os trabalhos de [Feamster et al. 2014, Braun and Menth 2014, Sezer et al. 2013, Macedo et al. 2015], o paradigma vem ganhando a atenção nos últimos anos porque, além de abordar a falta de programabilidade em arquiteturas de rede existentes, permitiu a inovação na rede de uma forma mais fácil e rápida, através de sua ideia básica, que é a separação dos planos de controle (que decide como manipular o tráfego) e de dados (que manipula o tráfego de acordo com as decisões estabelecidas pelo plano de controle). Dessa forma, novas pesquisas e desenvolvimentos tecnológicos se tornaram possíveis em redes em produção, sem que se afetasse o tráfego ou a disponibilidade da mesma. Assim, diversos serviços de rede estão sendo repensados, de forma a torná-los mais flexíveis.

O paradigma SDN atraiu a atenção da indústria e da comunidade acadêmica, e parte dessa atenção se voltou para o protocolo OpenFlow. O OpenFlow [McKeown et al. 2008] foi definido como um protocolo aberto para a programação da tabela de fluxos de *switches* e roteadores, o que passou a permitir o particionamento do tráfego da rede de acordo com os interesses de seus operadores. O protocolo ainda pode ser definido como uma API que interconecta os equipamentos de rede (plano de dados) a um controlador (plano de controle) [Feamster et al. 2014, Sezer et al. 2013]. O OpenFlow acabou se tornando a plataforma SDN de referência, utilizada na maioria dos desenvolvimentos e em pesquisas envolvendo o paradigma [Macedo et al. 2015]. E com o crescente interesse da indústria no OpenFlow, cada vez mais equipamentos de rede, como *switches* e roteadores, de empresas como HP, NEC, Pronto, Extreme, Cisco, Brocade, Juniper e Huawei, por exemplo, passaram a suportar o protocolo [Feamster et al. 2014].

Uma série de pesquisas relacionadas à SDN e ao padrão OpenFlow foi realizada

nos últimos anos. Contudo, ainda são raros os trabalhos que enfoquem o plano de dados dos *switches* OpenFlow, bem como suas implementações e mecanismos internos.

Alguns trabalhos abordam, de certa forma, a temática do plano de dados. Entretanto, tais trabalhos possuem abordagem e objetivos diferentes dos nossos. Por exemplo, [Bianco et al. 2010] analisam o desempenho de um único *switch*, implementado virtualmente sobre uma ponte (*bridge*). Os autores comparam o desempenho desse *switch* em modo OpenFlow, com o desempenho do mesmo operando através de roteamento IP e de *switching* Ethernet (modo *legacy*). Ou seja, eles comparam técnicas de encaminhamento distintas, utilizando o mesmo equipamento. Esse trabalho apresenta métricas de desempenho bem definidas, com foco em vazão de encaminhamento e latência de pacotes, sob condições normais e de sobrecarga, com diferentes padrões de tráfego e diferentes tamanhos de pacotes na rede.

O trabalho de [Sünnen 2011] também avalia o desempenho de um único *hardware switch* OpenFlow. Algumas das métricas de avaliação dependem fortemente do desempenho do controlador, que não é foco de estudo do nosso trabalho.

Por sua vez, [Appelman and De Boer 2012] realizam uma comparação entre diferentes *switches* OpenFlow. Foram comparadas três plataformas OpenFlow, sendo dois *hardware switches* e uma versão em *software* do Open vSwitch. Contudo, a avaliação contou fortemente com a participação do controlador, o que influenciou de forma considerável as avaliações conduzidas.

Finalmente, [Rotsos et al. 2012] propõem um framework aberto e genérico que permite o teste de *switches* OpenFlow. Eles desenvolveram uma ferramenta que testa a interação entre o mecanismo de encaminhamento do *switch* e a aplicação de controle remota, tanto para *hardware switches* quanto para *software switches*. O trabalho apresenta a ferramenta, assim como avalia algumas implementações de *switches* OpenFlow. Assim como nos trabalhos supracitados, a participação do controlador não permite que os resultados sejam conclusivos sobre o desempenho único e exclusivo dos *switches*.

Dessa forma, destacamos os seguintes diferenciais do nosso trabalho: (i) Comparamos diferentes implementações OpenFlow de *hardware switches* comerciais e de *software switches open source*. Essa comparação é realizada através de métricas de desempenho tais como latência e *jitter*. (ii) Analisamos apenas aspectos referentes ao plano de dados dos *switches*, descartando a influência do controlador. (iii) Comparamos os desempenhos de cada *switch* em seus modos normal e OpenFlow. (iv) Indicamos quando é vantajoso ou não utilizar o protocolo OpenFlow nos *switches* analisados. Em parte, nossos resultados nos permitem avaliar se as implementações do protocolo são feitas em *hardware*, ou se são instâncias de um *software switch* adaptadas ao mecanismo interno do *switch*.

3. Metodologia de Avaliação

Topologia de avaliação. Foi montada uma topologia típica SDN para as avaliações e testes de comparação das diferentes implementações de *switches* OpenFlow, como mostra a Figura 1. De forma geral, a topologia é composta por três máquinas interligadas através de um *switch* OpenFlow. Uma das máquinas atua como cliente, uma como servidor e a terceira como controlador da rede. Montamos essa topologia de modo que em

cada cenário de teste apenas o *switch* fosse variado. De fato, utilizamos máquinas e rede dedicadas aos experimentos, assim como disponibilizamos a mesma largura de banda em todos os testes, de modo que houvesse o mínimo de impacto possível causado pela rede ou pela configuração das máquinas. Como as aplicações executadas nos computadores são extremamente leves, controlador, cliente e servidor não operaram em saturação total. Também realizamos testes prévios com a vazão de cada *switch* e obtivemos valores máximos de vazão muito semelhantes entre si, de tal modo que poderíamos descartar a influência da vazão nos testes.

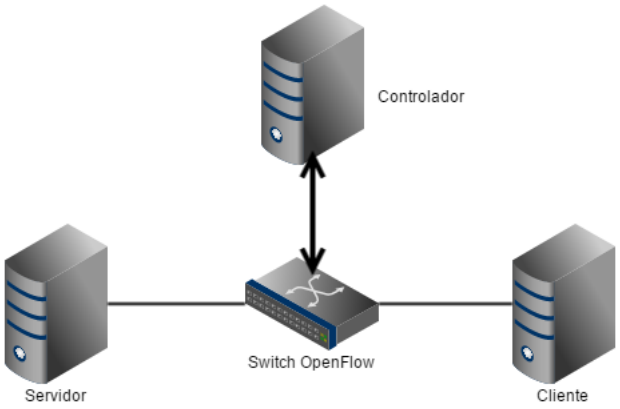


Figura 1. Arquitetura da rede de avaliação.

Para instalar as regras OpenFlow de avaliação, utilizamos o controlador POX [POX 2015]. O controlador instala regras com *hard timeout* ilimitado, no momento da inicialização da rede. As regras ficam em atividade até o momento em que o *switch* é desconectado. Nos testes com *switches* operando em modo *legacy*, sem estar ativo o protocolo OpenFlow, o controlador não realiza nenhuma operação.

***Switches* avaliados.** Com o surgimento do OpenFlow e sua grande aceitação no mercado, várias empresas passaram a produzir equipamentos com suporte a OpenFlow, seja através de prototipação, de atualização de *firmware* ou do desenvolvimento direto dessa tecnologia [Rotsos et al. 2012]. Neste trabalho serão avaliadas e comparadas algumas dessas implementações OpenFlow para *hardware switches* comerciais e *software switches* conhecidos, considerando a versão 1.0 do OpenFlow. A Tabela 1 apresenta os *switches* avaliados e sumariza suas características.

Caracterização da carga. Os testes consistem no envio de pacotes UDP entre a máquina cliente e o servidor através do switch OpenFlow a ser avaliado. Para isso, foi implementado um gerador de tráfego próprio, composto por dois módulos: um módulo instalado na máquina cliente e outro no servidor. O módulo cliente envia os pacotes UDP para o servidor, com o *timestamp* de envio do pacote. O módulo executado no servidor, ao receber os pacotes, os retorna para o cliente. O módulo cliente, ao receber a resposta, calcula o Round Trip Time (RTT) de cada um dos pacotes utilizando os *timestamps* de envio e de chegada. A partir disso, o módulo cliente pode computar as métricas de interesse, como atraso e *jitter* dos pacotes. Vale ressaltar que a precisão dos *timestamps* está na casa dos microssegundos, a mais exata possível, segundo a RFC 3161.

Tabela 1. *Switches* avaliados no trabalho.

Marca	Modelo	S.O. Firmware	HW / SW Switch	Núm. de portas	CPU
Extreme	**Summit x460-24p**	ExtremeXOS 15.4.2.8	HW	28	Single Core CPU 500 MHz
Extreme	**Summit x440-48p**	ExtremeXOS 15.4.2.8	HW	52	Single Core CPU 500 MHz
HP	**HP2920-24G**	Firmware K 15.5 i	HW	24	Tri Core ARM1176 625 MHz
LinkSys	**WRT54GL**	OpenWRT Pantou	SW	4	Broadcom BCM5352 200 MHz
Open vSwitch	**OvS 2.3.0**	Linux Ubuntu 14.04	SW	-	Intel Core i7 CPU 2.80 GHz

Variamos o tamanho dos pacotes UDP entre 64, 128 e 256 bytes, para avaliar o desempenho dos *switches* com diferentes *overheads* de processamento de dados. Esses valores foram escolhidos com base nos trabalhos relacionados e na RFC 2544. Os resultados de cada teste correspondem ao envio de 10.000 pacotes, e as curvas apresentam intervalos de confiança com 99%.

Foram definidos quatro cenários para a avaliação de desempenho das implementações OpenFlow de cada *switch*. Para cada um deles, foi elaborado um tipo de teste diferente, de acordo com as especificidades de cada caso.

4. Resultados

Nesta seção apresentaremos os resultados mais significativos dos experimentos realizados nos diferentes *switches* com suporte a OpenFlow.

4.1. Desempenho em diferentes tipos de match

O primeiro cenário definido avalia o desempenho do *switch* na realização de diferentes tipos de casamento (ou *match*). Definimos cinco tipos genéricos de *match*, compostos por subconjuntos de matches individuais formados dentre os 12 atributos da versão 1.0 do OF (Tabela 2). O objetivo é verificar se o tempo de execução dos *matches* é influenciado pelos diferentes atributos, e, principalmente, analisar a diferença do desempenho dos mesmos tipos de *match* em diferentes *switches*.

Tabela 2. Subconjuntos de atributos utilizados para *matches*.

Tipo de Match	Atributos de Match OpenFlow
Porta	Porta do *switch*
MAC	Endereços MAC de origem e destino
IP	Ethertype e endereços IP de origem e destino
Porta UDP	Ethertype, endereços IP de origem e destino, protocolo da camada de transporte e portas TCP/UDP de origem e destino
Exato	Porta do *switch*, endereços MAC de origem e destino, ethertype, ID da VLAN, prioridade da VLAN, endereços IP de origem e destino, protocolo da camada de transporte, IP ToS e portas TCP/UDP de origem e destino

Na Tabela 3 listamos os intervalos de confiança dos atrasos médios de pacotes de 64 bytes, obtidos na realização de cada um dos diferentes tipos de *match*, em cada

switch. Podemos verificar que em cada *switch* as diferentes operações de *match* possuem intervalos de atraso muito próximos um dos outros. Isso leva a crer que o tipo ou a quantidade de atributos utilizados para realizar um *match* não afetam substancialmente o atraso dos pacotes, quando comparamos os diferentes tipos de *match*. A única exceção ocorreu na realização do *match* exato no *switch* HP. O intervalo de confiança desse *match* no *switch* HP não foi coincidente com o dos demais tipos de *match*. O tempo de atraso médio foi destacadamente menor, quando comparado aos demais *matches*. Ainda nesta seção, apresentaremos uma possível explicação para isso.

Tabela 3. Intervalo de confiança do atraso para diferentes tipos de match (em ms).

Switch	Tipos de Match				
	Porta	**MAC**	**IP**	**Porta UDP**	**Exato**
Extreme x460-24p	0.125 - 0.127	0.124 - 0.126	0.124 - 0.126	0.124 - 0.126	0.121 - 0.125
Extreme x440-48p	0.174 - 0.175	0.131 - 0.133	0.130 - 0.132	0.119 - 0.121	0.117 - 0.119
HP	1.069 - 1.147	1.049 - 1.113	1.064 - 1.309	1.038 - 1.106	0.743 - 1.001
LinkSys - OpenWRT	1.025 - 1.051	1.017 - 1.045	1.019 - 1.049	1.017 - 1.046	1.018 - 1.047
Open vSwitch	0.514 - 0.519	0.502 - 0.508	0.504 - 0.508	0.513 - 0.517	0.508 - 0.511

Ao compararmos o mesmo tipo de *match* em diferentes *switches*, notamos uma substancial diferença entre os intervalos de confiança. Conseguimos notar três comportamentos distintos, para os tipos de *match* por porta, MAC, IP e porta UDP. Os *switches* Extreme (x460 e x440) apresentam os menores valores de atraso médio de todos os *switches* comparados, com aproximadamente 0.1 ms de atraso médio. Por sua vez, o Open vSwitch apresenta um valor médio de atraso próximo a 0.5 ms. Já o *switch* HP e o OpenWRT Pantou rodando no LinkSys WRT54GL [Pantou 2015] possuem os piores desempenhos, com os intervalos de confiança do atraso beirando 1 ms. Para ilustrar essa situação, o gráfico da Figura 2 apresenta as distribuições cumulativas (CDF) dos atrasos dos pacotes em cada *switch*, para o *match* por IP. Vale ressaltar que esse comportamento é observado também nas curvas dos *matches* por porta, MAC e porta UDP. No gráfico, é possível notar a diferença entre os três grupos formados a partir das distribuições. Os valores médios do atraso do *switch* HP e LinkSys OpenWRT são fortemente influenciados por atrasos de pacotes superiores a 2 ms.

Contudo, para o *match* exato, notamos uma pequena diferença. Como dissemos anteriormente, o *switch* HP possuía um atraso médio menor quando comparado ao dos outros tipos de *match*. Observando o gráfico da Figura 3, com a distribuição cumulativa dos atrasos dos pacotes para o *match* exato, notamos que a curva correspondente ao *switch* HP fica deslocada mais à esquerda, se aproximando da curva do Open vSwitch. Isso significa que os tempos de atraso dos pacotes nesse *match*, para o *switch* HP, são relativamente menores quando comparados aos atrasos nos outros *matches* no mesmo *switch*, em cerca de 21%. Acreditamos que isso se deve a algum mecanismo do *switch* HP que realiza *matches* exatos possivelmente em *hardware*, de forma mais rápida, ao contrário dos *matches* não exatos, que devem ser realizados em *software*, de forma mais lenta. Mesmo assim, para o *match* exato, os *switches* Extreme continuaram como os mais rápidos e a implementação OpenWRT do LinkSys como o mais lento dos *switches* analisados.

Para testarmos a influência do tamanho do pacote nos *matches*, realizamos os mesmos testes variando o tamanho dos pacotes para 128 e 256 Bytes. Quando comparamos

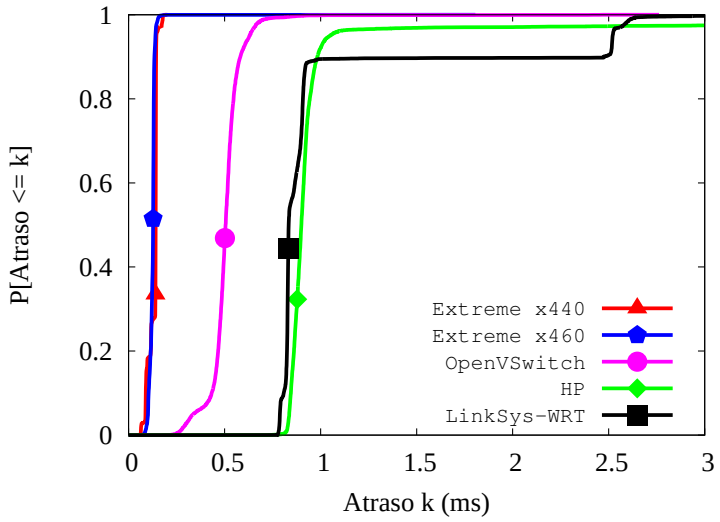


Figura 2. Distribuições cumulativas dos atrasos para o tipo de *match* por IP.

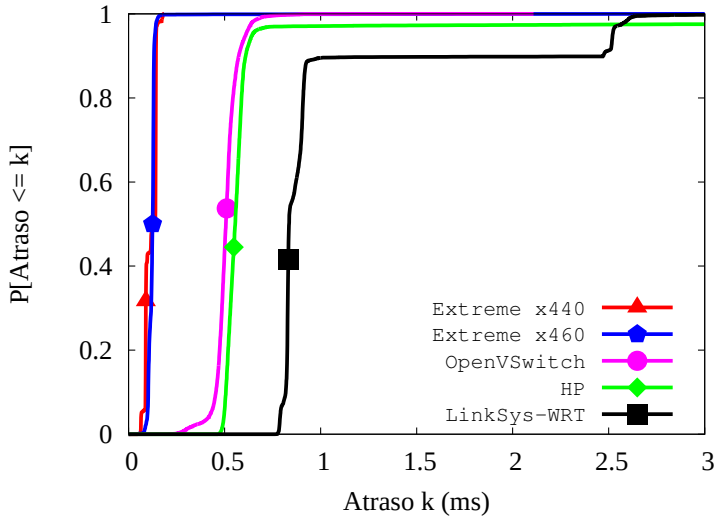


Figura 3. Distribuições cumulativas dos atrasos para o tipo de *match* exato.

os resultados para tamanhos diferentes de pacote, notamos que os *switches* HP e Extreme x460 tiveram um aumento maior nos tempos de atraso. Os outros não tiveram aumentos significativos nos atrasos. Podemos notar essa situação ao comparar o gráfico da Figura 4 (distribuições cumulativas dos atrasos para *matches* exatos com pacotes de 256 bytes) com o gráfico da Figura 3 anterior (com pacotes de 64 bytes). O deslocamento das curvas referentes aos *switches* Extreme x460 e HP é perceptível entre os gráficos.

O gráfico da Figura 5 mostra a CDF dos *jitters* de pacotes em *matches* exatos. Observamos que os *jitters* dos *switches* Extreme foram notoriamente menores quando comparados aos outros, ficando abaixo de 1 ms. Por sua vez, a curva do *switch* HP se assemelha às curvas dos *software switches* estudados. Isso pode corroborar a nossa suspeita de que a implementação OpenFlow do *switch* HP é apenas uma instanciação de um *software switch* rodando em memória, dentro de um *hardware switch*.

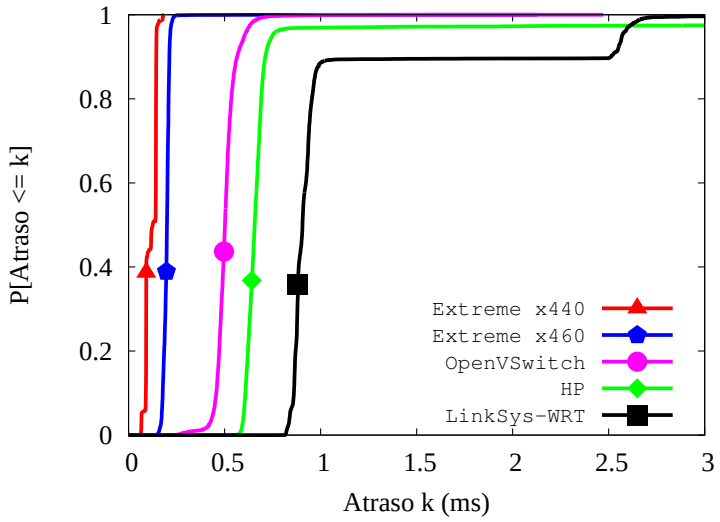


Figura 4. Distribuições cumulativas dos atrasos para o tipo de match exato em pacotes de 256 bytes.

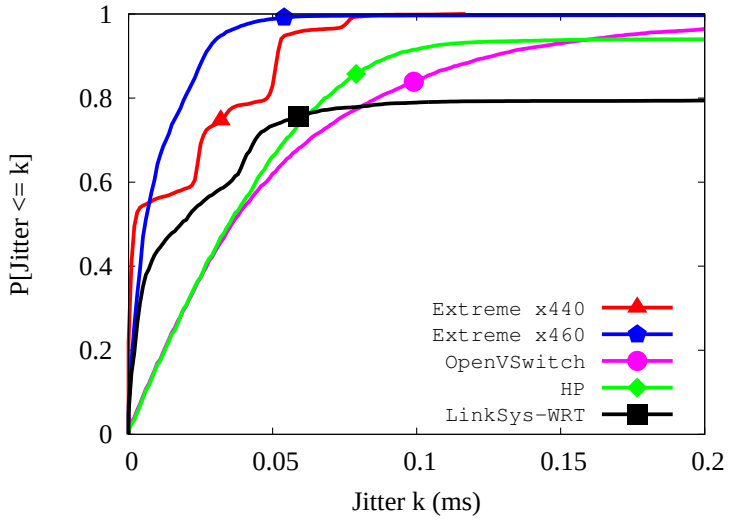


Figura 5. Distribuições cumulativas dos jitters para pacotes de 64 bytes.

4.2. Desempenho dos *switches* nos modos *legacy* e OpenFlow

O segundo cenário definido compara os desempenhos de um determinado *switch* quando operando nos modos OpenFlow e normal (*legacy*). Comparamos o encaminhamento dos pacotes entre a porta de chegada e a porta de saída do *switch* nos dois modos de operação, para verificar se existe alguma diferença de desempenho entre os modos.

Para reproduzir o encaminhamento porta a porta de um *switch* normal no modo OpenFlow, instalamos previamente uma regra no controlador que realizava *matches* no atributo de porta de entrada (*in_port*) e repassava os pacotes que sofriam esse *match* para uma porta de saída determinada. Só assim seriamos capazes de comparar os dois modos de operação de forma igualitária.

No gráfico da Figura 6 mostramos o desempenho dos *switches* em modo

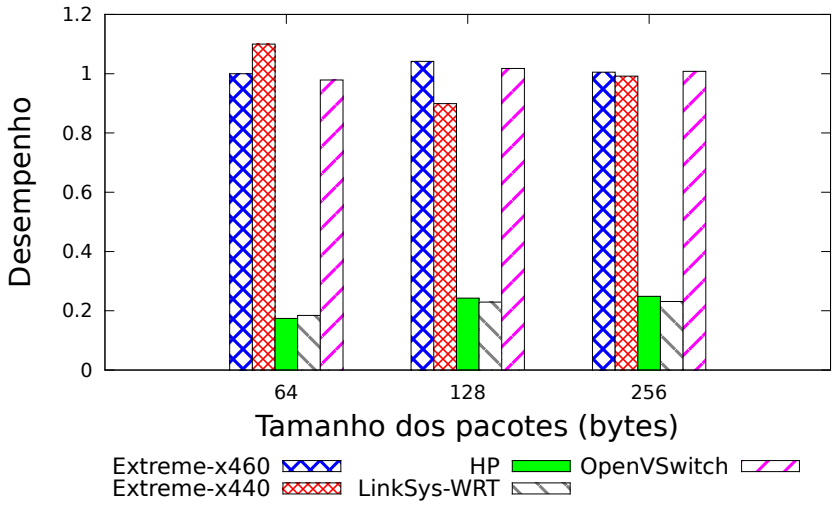


Figura 6. Desempenho dos switches em modo OpenFlow em comparação ao modo legacy.

OpenFlow quando comparados ao desempenho dos mesmos em modo normal (*legacy*), para três tamanhos diferentes de pacotes. A comparação entre os desempenhos foi obtida através da métrica de *Speedup*, que consistiu na divisão da latência no modo normal pela latência no modo OpenFlow. No gráfico, valores próximos a 1.0 correspondem a desempenhos semelhantes dos modos OpenFlow e normal. Valores menores que 1.0 indicam uma piora no desempenho no modo OpenFlow, enquanto valores superiores a 1.0 indicam um desempenho superior utilizando OpenFlow. Os intervalos de confiança são pequenos, por isso não estão reproduzidos na imagem.

Ao analisar o gráfico, nota-se que o desempenho do modo OpenFlow é bem próximo ao desempenho do modo *legacy* nos dois *switches* Extreme (x460 e x440) e no Open vSwitch. Já os desempenhos do modo OpenFlow nos *switches* HP e LinkSys-OpenWRT são bem inferiores aos desempenhos em modo *legacy*, chegando a apenas 20% do desempenho normal em ambos, aproximadamente. Essas observações nos levam a crer que já é vantagem, em alguns equipamentos, utilizar a tecnologia OpenFlow. Nesses casos, a utilização do OpenFlow praticamente não acarretou sobrecarga em operações normais do *switch*. Além disso, é possível pensar que os mecanismos de encaminhamento internos desses *switches* podem também ser utilizados pela tecnologia OpenFlow, o que explicaria um desempenho semelhante entre os dois modos.

4.3. Desempenho com uma ou muitas regras na tabela de fluxos

O terceiro cenário definido avalia o desempenho do *switch* em duas situações: quando a tabela de fluxos possui apenas uma regra, e esta retorna um *match*; e quando a tabela possui múltiplas regras, mas apenas uma delas retorna um *match*. Nosso objetivo é verificar se o desempenho pode ser influenciado pela disposição das regras na tabela de fluxos, pelo processamento interno de cada *switch* ou pelo tipo de regra instalado.

O primeiro passo para realizar esse teste foi a descoberta do limite máximo de regras que poderíamos instalar em cada *switch* no modo OpenFlow. A Tabela 4 apresenta o máximo de regras instaladas nos experimentos, além do número máximo de regras

suportadas pelos *switches* em modo *legacy*. Ao analisar esses dados, notamos que, na maioria dos casos, o número de regras suportadas no OpenFlow é menor do que o número de regras suportadas nativamente.

Tabela 4. Quantidade de regras suportadas pelos *switches* nos modos *legacy* e OpenFlow.

Switches	Quantidade de Regras Suportadas no Modo *Legacy*	Quantidade de Regras Observadas no Modo OpenFlow
Extreme x460-24p	2048	1200
Extreme x440-48p	1024	248
HP	2048	16000
LinkSys - OpenWRT	100	100
Open vSwitch	1000000	750000

Ao investigar o motivo dessa diferença, notamos que os *switches* Extreme (x460 e x440) armazenam as regras OpenFlow em formato de *access-lists* (ACLs) em *hardware*, assim como fazem com as regras quando operando em modo normal. Isso pode explicar o desempenho semelhante dos dois modos de operação (OpenFlow e *legacy*) para os *switches* Extreme, mostrados no cenário anterior. Como, no modo OpenFlow, existem mais alguns atributos ou operações a serem armazenadas, e o tamanho da ACL é definido como *double*, menos regras são suportadas no modo OpenFlow. Já o Open vSwitch possui capacidade de armazenamento de até 1 milhão de regras em uma tabela *hash* desenvolvida para armazenar apenas *matches* exatos, mas escolhe o limite de acordo com os tipos de regras e as condições da rede. Para armazenar regras de matches não exatos, o Open vSwitch utiliza uma tabela linear de até 100 posições.

Em relação ao *switch* HP, na instalação da tecnologia OpenFlow, ele passa a possuir além de uma tabela em *hardware*, uma outra tabela em *software*, com uma grande capacidade de armazenamento de regras, e que não pode ser desativada. Esse fato pode indicar que, possivelmente, essa implementação do OpenFlow seja uma implementação em *software* do Open vSwitch, sem maior integração ao projeto de *hardware* do *switch* HP. Isso explicaria o baixo desempenho no modo OpenFlow do *switch* HP, apresentado no cenário anterior.

Tabela 5. Intervalos de confiança dos atrasos em tabelas com uma e múltiplas regras (em ms).

Switches	Uma regra	Múltiplas regras
Extreme x460-24p	0.121 - 0.125	0.124 - 0.128
Extreme x440-48p	0.109 - 0.110	0.114 - 0.115
HP	1.049 - 1.120	1.046 - 1.330
LinkSys - OpenWRT	1.028 - 1.055	1.215 - 1.242
Open vSwitch	0.508 - 0.511	0.513 - 0.517

Na Tabela 5 estão listados os intervalos de confiança do atraso médio dos pacotes em duas situações: no caso em que a tabela de fluxos contém apenas uma regra, e no caso em que a tabela de fluxos possui múltiplas regras falsas e apenas uma regra como verdadeira. Em ambos os casos, os *matches* foram exatos. A quantidade de regras instaladas, no último caso, equivale ao número limite de regras encontrado no experimento anterior, listado na Tabela 4 anteriormente. Vale ressaltar que nos switches com regras escritas

em ACLs, a regra verdadeira foi inserida por último na tabela de fluxos. Já nos switches baseados em *software*, como as regras são inseridas utilizando índices *hash*, é impossível determinar sua efetiva posição na tabela de fluxos.

Podemos notar que, na maioria dos *switches*, os intervalos se interceptam ou apresentam diferença de poucos microssegundos entre si. Isso pode indicar que a grande quantidade de regras instaladas e a disposição das mesmas na tabela de fluxos não sobrecarregam significativamente os *switches*, nem causam um sensível aumento nos atrasos. A única exceção a essa observação ocorre com o LinkSys-OpenWRT, que possui um aumento médio no atraso dos pacotes em torno de 0.2 ms.

4.4. Desempenho na modificação de campos do cabeçalho dos pacotes

O quarto e último cenário avalia o desempenho de cada *switch* em modificações de campos específicos do cabeçalho de cada pacote. No teste desse caso, foram realizadas operações de modificação em campos do cabeçalho dos pacotes como endereços MAC e IP de destino, prioridade de VLAN, IP *Type of Service* e porta UDP, com o objetivo de comparar essas operações nos diferentes *switches*. Vale ressaltar que as operações de modificação de cabeçalho são opcionais no OpenFlow 1.0.

Tabela 6. Intervalos de confiança dos atrasos médios em modificações dos campos de cabeçalho (em ms).

Tipo de modificação	Switches				
	Extreme x460	**Extreme x440**	**HP**	**LinkSys-OpenWRT**	**Open vSwitch**
Sem modificação	0.125 - 0.127	0.114 - 0.115	1.121 - 1.212	1.032 - 1.058	0.508 - 0.511
MAC	0.125 - 0.129	0.113 - 0.115	1.128 - 1.203	1.034 - 1.061	0.511 - 0.516
IP	-	-	1.107 - 1.178	1.034 - 1.060	0.510 - 0.515
Prioridade de VLAN	0.125 - 0.128	-	1.107 - 1.183	-	-
IP ToS	-	-	1.122 - 1.194	1.030 - 1.056	0.506 - 0.510
Porta UDP	-	-	1.071 - 1.132	1.033 - 1.060	0.512 - 0.515

Na tabela 6 estão listados os intervalos de confiança dos atrasos dos pacotes quando estes sofrem ações de modificação em cinco campos de seus cabeçalhos, individualmente. Para efeito de comparação, colocamos também os valores do caso em que os pacotes não sofrem nenhuma alteração em seus cabeçalhos. Para todos os casos, não se notou um considerável aumento nos tempos de atraso médio dos pacotes nas modificações, quando comparados ao caso em que não há modificação. Isso indica que a operação de modificação pode ocorrer na ordem de poucos microssegundos. Entretanto, notamos que ainda existe falta de suporte para algumas operações de modificação em alguns *switches*. De fato, essa limitação está prevista na documentação de cada um deles. Essa falta de suporte, aliada ao fato de que só é permitido realizar uma única operação de modificação de cabeçalho por regra, mostra que ainda existem aspectos a se melhorar nas implementações OpenFlow já existentes, sobretudo em *hardware*. Devido a certas configurações da arquitetura, algumas operações de modificação não puderam ser realizadas e não foram inseridas na tabela.

5. Conclusão

Neste artigo avaliamos o desempenho do plano de dados de *hardware* e *software switches* OpenFlow. O objetivo deste trabalho foi avaliar a maturidade das implementações existentes, dado que o OpenFlow já se tornou uma tecnologia com diversos usos em redes de produção, e o seu uso tende a aumentar nos próximos anos.

A avaliação levou em consideração três *hardware switches* comerciais, e dois *software switches open source*. Foram avaliados o atraso de entrega de pacotes, o desempenho dos *switches* com diferentes tipos de regras de *match* de cabeçalhos, e se houve uma diferença de desempenho dos *switches* em modo OpenFlow em relação a um *switch* L2 convencional.

Os resultados mostraram que os *switches* implementam OpenFlow de formas diferentes. Alguns o implementam em *hardware*, outros em *software*. Além disso, a forma como a implementação foi realizada afeta significativamente o desempenho dos *switches*.

Como trabalhos futuros, pretendemos expandir a lista de *switches* avaliados, estudando alguns *switches* muito empregados pela comunidade de pesquisa Brasileira e do mundo. Por exemplo, pretendemos avaliar os *switches* Pica8, Datacom e NetFPGA (utilizados nas ilhas FIBRE[1]).

Referências

Appelman, M. and De Boer, M. (2012). Performance analysis of openflow hardware. Semester thesis project report, University of Amsterdam.

Bianco, A., Birke, R., Giraudo, L., and Palacin, M. (2010). Openflow switching: Data plane performance. In *IEEE International Conference on Communications*, pages 1–5.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and Walker, D. (2014). P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95.

Braun, W. and Menth, M. (2014). Software-defined networking using openflow: Protocols, applications and architectural design choices. *Future Internet*, (6):302–336.

Feamster, N., Rexford, J., and Zegura, E. (2014). The road to SDN: an intellectual history of programmable networks. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 87–98. ACM New York, NY, USA.

Ibarra, J., Bezerra, J., Morgan, H., Lopez, L. F., Cox, D. A., I., Stanton, M., Machado, I., and Grizendi, E. (2015). Benefits brought by the use of openflow/sdn on the amlight intercontinental research and education network. In *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, pages 942–947.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., and Vahdat, A. (2013). B4: Experience with a globally-deployed software defined wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 3–14, New York, NY, USA. ACM.

[1]https://fibre.org.br/

Khondoker, R., Zaalouk, A., Marx, R., and Bayarou, K. (2014). Feature-based comparison and selection of software defined networking (sdn) controllers. In *2014 World Congress on Computer Applications and Information Systems, WCCAIS 2014*, pages 1–7.

Kreutz, D., Ramos, F., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Macedo, D. F., Guedes, D., Vieira, L. F. M., Vieira, M. A. M., and Nogueira, M. (2015). Programmable networks-from software-defined radio to software-defined networking. *IEEE Communications Surveys and Tutorials*, 17(2):1102–1125.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. In *SIGCOMM Comput. Commun. Rev.*, volume 38, pages 69–74, New York, NY, USA.

ONF (2015). What is ONF? Disponível em: https://www.opennetworking.org /images/stories/downloads/about/onf-what-why.pdf. Acesso: novembro/2015.

Pantou (2015). Pantou : OpenFlow 1.0 for OpenWRT. Disponível em: http://archive.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT. Acesso: novembro/2015.

POX (2015). Pox wiki. Disponível em: https://openflow.stanford.edu/display/ONL/POX +Wiki. Acesso: novembro/2015.

Rotsos, C., Sarrar, N., Uhlig, S., Sherwood, R., and Moore, A. W. (2012). *OFLOPS: An open framework for OpenFlow switch evaluation*, volume 7192 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Sezer, S., Scott-Hayward, S., Chouhan, P.-K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. In *IEEE Communications Magazine*, volume 51, pages 36–43.

Sünnen, D. (2011). Performance evaluation of openflow switch. Semester thesis project report, Swiss Federal Institute of Technology Zurich.

Song, H. (2013). Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 127–132, New York, NY, USA. ACM.

Tennenhouse, D., Smith, J., Sincoskie, W., Wetherall, D., and Minden, G. (1997). A survey of active network research. *Communications Magazine, IEEE*, 35(1):80–86.

Trilha Principal do SBRC 2016
Sessão Técnica 27
Redes Ópticas e Roteamento

Roteamento e Alocação de Espectro Ciente da Aplicação em Redes Ópticas Elásticas

Léia S. de Sousa, Lucas R. Costa, Felipe R. de Oliveira, André C. Drummond, Eduardo Adilio Pelinson Alchieri

[1]Departamento de Ciência da Computação (CIC) – Universidade de Brasília (UnB)
Caixa Postal 15.064 – 70910-900 – Brasília – DF – Brasil

Abstract. *Based on the Cross-Layer Design (CLD) paradigm, this paper presents a application-aware routing and spectrum allocation (AA-RSA) solution of Multiple Bulk Data Transfer (MBDT), services which transports massive data amount in a geo-distributed network, taking into account their data traffic pattern to improve service to the demands of resynchronization in networks serving data centers (DC), or inter-data center network (IDC).*

Resumo. *Baseado no paradigma Cross-Layer Design (CLD), este trabalho apresenta uma solução de roteamento e atribuição de espectro ciente da aplicação (AA-RSA) de Múltiplas Transferências de Dados em Massa (MBDT), que são serviços de transporte de volumosas massas de dados em uma rede geo-distribuída. O algoritmo proposto leva em consideração o padrão de tráfego de dados para melhorar o atendimento às demandas de ressincronização em redes utilizadas por centros de dados (CD), ou redes inter-centro de dados (ICD).*

1. Introdução

Baseado no paradigma *Cross-Layer Design* (CLD), que rompe com o conceito de camadas isoladas promovendo a intercomunicação vertical entre camadas, o roteamento ciente da aplicação (*Application-Aware Routing*, AA-R) se destaca do roteamento convencional por explorar informações a respeito das características de cada tipo de aplicação, como *video streaming*, jogos *online* e de transferências de dados, e aplicá-las na melhoria do desempenho da rede e qualidade dessas aplicações [Gerber and Doverspike 2011], [Subramaniam et al. 2013].

As Múltiplas Transferências de Dados em Massa (MBDTs) são um exemplo de aplicação que, por definição, movimentam volumosas quantidades de dados ao longo de uma rede de centro de dados (CD) geo-distribuída, como forma de alcançar melhor desempenho e maior confiabilidade. São implementadas nos ambientes de computação em nuvem para replicar e sincronizar as bases de conteúdos de grandes provedores desse tipo de serviço [Laoutaris et al. 2009]. Hoje, a alocação de recursos para MBDT é estática com taxas previamente definidas, onde uma operação completa de transferência pode levar de dias a semanas, o que é feito tipicamente nos momentos de menor utilização da rede (padrão noturno) e leva os provedores a utilizarem armazenamento de dados em centros de dados alugados durante o dia, sendo que a rede não é ciente das aplicações que são executadas; além disso, não existem técnicas de engenharia de tráfego que garantam as transferências dentro de um específico período de tempo (*deadline*) [Zhang et al. 2015].

O tráfego elástico das MBDT pode ser paralisado e reiniciado conforme prioridade definida pelos *Internet Service Providers* (ISPs). Os canais utilizados são *backbones* de

redes ópticas de alta velocidade, que podem ser linhas proprietárias (ex: *Google*, *Yahoo!* e *Microsoft* [Ghosh et al. 2013]) ou públicas (ex: a *Internet* [Gerber and Doverspike 2011]).

Para atender às expectativas futuras, esses *backbones* poderão implementar o sistema de comunicação óptico elástico (EON). A EON utiliza menores espaçamentos de grades do espectro óptico e transporta mais *bits* por símbolos, o que permite transportar mais dados com menos recursos e melhor ajuste das demandas às larguras de banda [Christodoulopoulos et al. 2011], [Jinno et al. 2009]. Os *transponders* de largura de banda flexível (BVTs) da arquitetura EON, equipamentos que convertem o sinal vindo da rede cliente para um sinal que possa ser comutado no domínio óptico, implementam técnicas *Orthogonal Frequency-Division Multiplexing*(OFDM) capazes de alocar *slots*, ou fatias do espectro óptico, para um determinado caminho de acordo com a taxa de transmissão solicitada. Os *cross-connects* de largura de banda flexível (BV-WXCs) dessa arquitetura estabelecem os caminhos ópticos com os recursos solicitados e podem suportar uma infinidade de caminhos com as mais variadas capacidades [Christodoulopoulos et al. 2011].

Encontrar uma rota e atribuir recursos de espectro com o mais apropriado formato de modulação para acomodar uma demanda constitui o problema fundamental em EON, chamado de problema de roteamento, modulação e atribuição de espectro (RMSA). Tal problema está sujeito às restrições de continuidade do espectro (todos os enlaces do caminho devem utilizar os mesmos *slots*), de contiguidade do espectro (os *slots* que atendem uma demanda devem ser consecutivos), de não sobreposição (duas demandas não podem utilizar simultaneamente o mesmo *slot*), e de banda de guarda (separação entre os *slots* de demandas diferentes). A versão com modulação fixa do problema RMSA é o roteamento e atribuição de espectro (RSA). Ambos são NP-completos [Wan et al. 2012].

Este trabalho propõe um esquema de roteamento em EON, que, a partir de informações recebidas da camada de aplicação, atende um número maior de conexões, aumentando o desempenho do sistema. Para mostrar os ganhos advindos com um esquema de roteamento ciente da aplicação, é definido um cenário de ressincronização ICD, no qual um CD volta a fazer parte da rede após um período de indisponibilidade e precisa atualizar as informações armazenadas (seu estado). Para isso, seus pares encaminham grandes volumes de dados com as informações atualizadas para que ocorra esta ressincronização.

Este cenário reflete a execução de um sistema distribuído, cujas informações são replicadas para garantir tolerância a falhas [Castro and Liskov 2002]. A quantidade de réplicas necessárias varia de acordo com o protocolo de replicação e o modelo de sistema adotado. Neste trabalho, consideramos o modelo mais genérico de falhas onde são necessárias 4 réplicas para que uma delas possa falhar [Castro and Liskov 2002]. Embora este número possa ser aumentado para permitir que mais réplicas falhem, na prática utiliza-se poucas réplicas [Vukolic 2010]. Desta forma, durante a ressincronização, um lote de comunicação de muitos para um ($M \rightarrow 1, M \geq 3$) é estabelecido. Na solução proposta, emprega-se uma abordagem de transferência fim-a-fim, sem armazenamento intermediário.

Os resultados obtidos indicam que o algoritmo ciente da aplicação estabelece cerca de 30% a mais de conexões para ressincronizações em comparação com o roteamento convencional, mantendo seu desempenho superior mesmo em condições de tráfego pesado na rede. Delegar mais responsabilidades para o roteamento aumenta o número de ressincronizações bem sucedidas, o que reduz as despesas com armazenamento em trânsito.

Neste sentido, as principais contribuições deste trabalho são: *(i)* proposta de um algoritmo de roteamento em EON ciente da aplicação de MBDT, chamado AA-RSA; e *(ii)* realização de uma série de simulações para avaliar o desempenho do algoritmo proposto face a um algoritmo RSA convencional, as quais demonstram claramente os ganhos advindos com a troca de informações entre as camadas de aplicação e de rede.

2. Trabalhos Relacionados

A partir do surgimento das pesquisas em EON [Jinno et al. 2009], soluções de roteamento e alocação de recursos vêm destacando o potencial elástico e dinâmico da tecnologia para melhorar a capacidade de transporte das redes, como [Wan et al. 2012] que propõe uma solução RSA com modulação fixa e outra que adapta modulações empregando níveis de modulação a partir do mais alto (com menor largura de banda e que transportam menos *bits* por símbolo) para satisfazer a mais demandas, sendo ambos roteamentos convencionais.

O paradigma CLD tem tradicionalmente relacionado as redes ópticas às pesquisas que tratam de *Physical Layer Impairments* (PLIs), mais especificamente *Impairment-Aware Routing and Wavelength Assignment* (IA-RWA) nas redes WDM, cujas soluções de roteamento apresentadas consideram as limitações da camada física da rede. Em [Zhao et al. 2015], por exemplo, limitações de não linearidade das fibras ópticas, que resultam em perdas nos sinais devido a mudanças de potência ao longo do caminho de propagação da luz em longas distâncias, bem como de defeitos de fabricação dessa fibra, são consideradas para melhorar o desempenho do roteamento. Já o roteamento ciente da aplicação foi explorado em [Song et al. 2012], onde propôs-se o mapeamento de máquinas virtuais aos seus respectivos *hosts* físicos para realizar migrações de acordo com a disponibilidade de recursos e com as distâncias entre os nós físicos, o que reduziu o tráfego de dados nos enlaces da rede e o consumo de energia.

Também motivado pela preocupação com o volume de tráfego, o problema MBDT foi destacado em [Li et al. 2012], com uma proposta para reduzir o tráfego inter-domínio utilizando os canais comerciais de ISPs associados a redes dedicadas, através das quais é imposto um esquema de escalonamento de dados em diferentes janelas de tempo. Em [Laoutaris et al. 2009], as MBDT são realizadas em horários noturnos. Também foi proposto o sistema *NetStitcher* [Laoutaris et al. 2011], capaz de identificar rapidamente as larguras de banda ociosas em qualquer enlace da rede geo-distribuída. Em comum, essas soluções propuseram variados modos de transferência aliados ao roteamento tradicional.

Em [Lu et al. 2015] e [Lu and Zhu 2015], são mostradas soluções para transferências de dados em massa (BDT) utilizando a tecnologia EON, as quais tentam aproveitar o espectro óptico fragmentado após o atendimento de requisições orientadas a fluxos para realizar as transferências orientadas a dados, embora não se tenha considerado aspectos dessas aplicações.

Este trabalho explora essa proposta de RSA com modulação fixa no âmbito do roteamento ciente das aplicações. Os resultados comparativos são apresentados para demonstrar o ganho de eficiência que pode ser alcançado quando a camada de rede possui mais informações para auxiliar a tomada de decisões.

3. Problema das Múltiplas Transferências de Massa de Dados na Ressincronização ICD

Um sistema distribuído capaz de tolerar falhas de *hardware* e até intrusões precisa realizar replicações geo-distribuídas dos seus dados. Cada nó de armazenamento desse sistema é responsável por um conjunto de informações, chamado de partição, que são replicados separadamente. Um grupo de replicação é um grupo de nós responsável pela mesma partição. Assim, cada CD pode participar simultaneamente de vários grupos diferentes. Os sistemas de gerenciamento de dados distribuídos (DMS) que lidam com esses CDs possuem um correto e consistente mapeamento dos grupos de replicação em seus membros e suas respectivas localizações [Sharov et al. 2015].

Várias rodadas de trocas de mensagens dentro de determinado grupo, que são definidas de acordo com cada protocolo específico, são realizadas quando CDs submetem requisições entre si. O pressuposto fundamental para que este mecanismo possa ocorrer é o estado comum das réplicas, alcançado mediante sincronizações e ressincronizações [Castro and Liskov 2002]. Um nó membro inativo que deseja voltar à rede após um período de indisponibilidade e cujo seu estado encontra-se desatualizado, pode submeter uma solicitação de integração a esse grupo e assim, acionar os serviços de ressincronização. O grupo de réplicas designado pelo DMS possui o mesmo estado e um mapeamento das partições a serem replicadas [Agrawal et al. 2013a]. Para atender a solicitação de ressincronização do nó que está retornando, o sistemas de gerenciamento de dados distribuídos define os respectivos nós candidatos capazes de atender a essa ressincronização e dispara as MBDTs dentro de um período de tempo suficiente para a completa atualização do nó solicitante.

Do ponto de vista da camada de rede, normalmente a aplicação MBDT é atendida juntamente com outras solicitações de roteamento, embora suas taxas de dados sejam muito variadas, sem distinção do tipo de tráfego ou do seu nível de prioridade, para as quais o roteamento estabelece um caminho com largura de banda disponível e move o tráfego fim-a-fim. Logo, embora as MBDTs sejam tolerantes ao atraso, algumas requisições podem não ser atendidas devido a indisponibilidade de recurso suficiente por um período estendido de tempo e, consequentemente, estrangulamento do seu *deadline* [Zhang et al. 2015].

A solução de roteamento em EON ciente da aplicação MBDT para ressincronização de CDs, AA-RSA, impõe maior complexidade por incorporar tomadas de decisões relacionadas a busca de combinações de um subconjunto dos nós aptos à transferência. Assim, o problema RSA tem seu espaço de busca ampliado de tal maneira que, além de tentar estabelecer caminhos ópticos com fatias espectrais suficientes, essas tentativas são experimentadas para todas os subconjuntos de 3 CDs transmissores que integram o grupo de replicação. Neste trabalho, consideramos grupos de transferências de 3 CDs porque o número de nós que compõem um grupo de ressincronização são geralmente 4 ou 5, sendo que as combinações de requisições consideram apenas os nós que farão transferências, que são 3 ou 4 [Vukolić 2010].

Esta solução é possível a partir da comunicação vertical com o DMS, que é encarregado de informar o tipo de tráfego a ser movido e de fornecer uma abstração da localização dos membros participantes do grupo de replicação, normalmente inacessível pelo roteamento. Essa troca de informações é garantida via CLD. A decisão por algum

subconjunto de replicação atende aos requisitos de tolerância a falhas da aplicação, garante melhor desempenho na ressincronização (mais requisições são aceitas) e reduz a utilização de recursos atendendo somente ao mínimo de transferências necessárias.

Para resolver o problema AA-RSA, a topologia física da EON foi modelada como um grafo direcionado $G(V, E)$, onde V e E denotam o conjunto de nós e enlaces da fibra, respectivamente. Assume-se que V é constituído por BV-WXCs e CDs geograficamente distribuídos e interconectados, representados como V^h. O conjunto V^h representa as localizações de CD e comutadores ópticos simultaneamente, e portanto, esse tipo de nó é capaz de solicitar e receber uma ressincronização. Cada enlace pode acomodar no máximo, BW *slots* de frequência do espectro.

Sobre essa rede é realizado o roteamento e alocação desses *slots* para determinar um caminho k com largura de banda B disponível a fim de transportar o tráfego ICD do tipo *Bulk Data Transfer* (BDT), que são chamadas orientadas a dados [Lu et al. 2015]. Uma requisição BDT é um *bulk*, modelado como $r_u = \{s_u, d_u, C_u, Dl_u\}$, onde $s_u \in V^h$ e $d_u \in V^h - \{s_u\}$ são a origem e o destino da chamada, C_u é a quantidade de dados a ser transferida e Dl_u é o *deadline* da transferência.

A requisição de ressincronização MBDT encaminhada do DMS e recebida no roteamento, é um conjunto de *bulks* r_u de tamanho n representado como $R = \{r_{u_1}, r_{u_2}, ..., r_{u_n}\}$, onde existe um conjunto S contendo todas as origens das transmissões e d e Dl são iguais para todas as requisições. Como principal característica deste tipo de aplicação, sua redundância inerente requer que a rede ICD seja altamente tolerante a falhas multi-nós. Por isso, um conjunto R_u deve ter uma cardinalidade mínima de 3 *bulks*. Com este conhecimento, as demandas de conexão terão semânticas que consideram a camada de aplicação, especificando o grupo de redundância participante, o que restringe o espaço de buscas por nós candidatos por parte da camada de rede. Devido a esta restrição, isto é, alocar recursos para as três transferências, o roteamento não tenta realizar alocações se a requisição viola esta condição.

Para exemplificar, a Figura 1 mostra um processo de ressincronização para o CD_5. Um lote MBDT com $S = \{CD_1, CD_2, CD_3, CD_4\}$ é enviado para o $d = \{CD_5\}$, movimentando um volume $C = \{1TB\}$, que deve ser recebido no destino dentro de um limite de tempo $Dl = \{30min\}$. Um algoritmo de roteamento convencional, como é o caso do RSA [Wan et al. 2012], adaptado para tratar de requisições orientadas a dados, considera esse lote como um conjunto de quatro chamadas independentes para as quais a busca por recursos é feita de maneira sequencial conforme a chegada dessas requisições. Nesse cenário, pode ocorrer de menos do que três chamadas serem atendidas e as demais serem bloqueadas devido ao esgotamento do tempo enquanto se aguardava por recursos de largura de banda ou mesmo devido a sua indisponibilidade nos enlaces compartilhados. Quando isso acontece, mesmo que algumas chamadas individuais sejam atendidas, não ocorre uma ressincronização com garantias de tolerância a falhas.

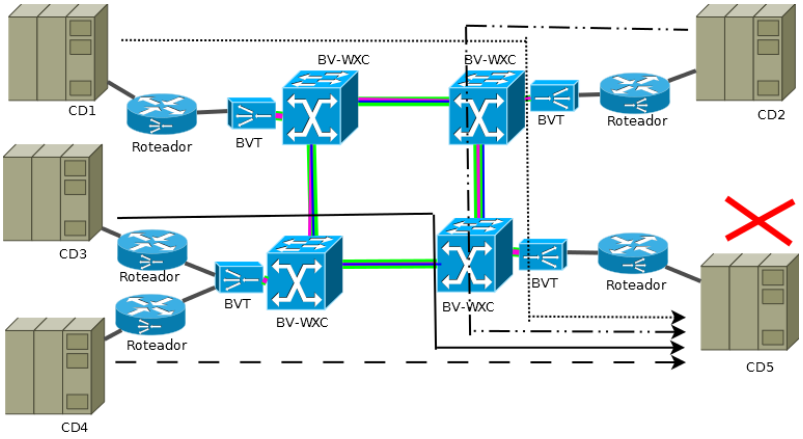


Figura 1. MBDT na ressincronização cross-DC

Ciente da importância dessa restrição para a aplicação, o roteamento pode esforçar-se em atender ao menos 3 *bulks* desse lote, limite inferior denominado fator de replicação, evitando bloqueá-las antes de ter certeza sobre a inexistência de recursos. Para isso, um algoritmo de combinações de lote pode acessar um lote e buscar por um subconjunto mínimo com essas características. Na Figura 1, as possíveis combinações seriam $\{\{CD1, CD2, CD3\}, \{CD1, CD2, CD4\}, \{CD2, CD3, CD4\}, \{CD1, CD3, CD4\}\}$. O AA-RSA propõe a busca por uma combinação mínima dos elementos de S. O atendimento de menos do que três demandas quebra a restrição de tolerância a falhas da aplicação de ressincronização de CDs. Por outro lado, atender mais do que 3 chamadas levaria ao desperdício de recursos.

4. Algoritmo AA-RSA

O algoritmo AA-RSA (Algoritmo 1) baseia-se no algoritmo RSA com modulação fixa [Wan et al. 2012]. Nas linhas 1 e 2 são inicializados o fator de replicação *b*, que representa a quantidade mínima de réplicas de CD para a ressincronização, e K, a quantidade de menores caminhos entre quaisquer dois nós. A função $COMBINATIONS(R_u, b)$, na linha 3, calcula todas as combinações possíveis de b CDs dentro do universo do lote R_u, que contém n chamadas para efetuar uma ressincronização. O seu processamento retorna uma matriz m de $\frac{n!}{b!(n-b)!}$ linhas e b colunas. Cada linha de m é um subconjunto de combinações *comb* de tamanho b. Em cada $comb$ executa-se para cada um de seus *bulks* i a função $KSP(i, K)$, linha 5, que retorna os K menores caminhos entre $s(i)$ e $d(i)$, origem e destino da chamada i, respectivamente, em ordem crescente de tamanho.

Na busca do *k*-ésimo caminho viável, nas linhas 6-13, para cada candidato $k \in K$ calcula-se a distância de k, linha 7, que é comparada com a fórmula $\mathscr{D} \times \tau$, onde $\mathscr{D}$ é o diâmetro da rede e τ é um parâmetro que restringe esse diâmetro a um limite de tamanho que permita utilizar a largura de banda máxima $MaxRate$ do caminho, equivalente à capacidade máxima de um BVT. Essa fórmula impede que requisições de conexões para os caminhos menores do que um determinado limite do maior caminho sejam penalizadas, com a alocação da mínima largura de banda disponível. A ideia é que recursos de caminhos relativamente pequenos sejam liberados mais rapidamente, desocupando recursos para as chamadas seguintes. Às requisições dos caminhos que não atendem a essa condição, na linha 10, é atribuída uma largura de banda B mínima, equivalente ao quociente da quantidade de dados transferida dentro do *deadline* total da sua chamada, $C_i \div Dl_i$.

Algoritmo 1 AA-RSA

1: $b \leftarrow 3$
2: $K \leftarrow 3$
3: $\mathbf{COMBINATIONS}(R_u, b)$
4: **for** $i = 1 \rightarrow b$ **do**
5: $\quad \mathbf{KSP}(i, K)$
6: $\quad$ **for** $k = 1 \rightarrow K$ **do**
7: $\quad\quad dist\,(k) = \sum_{l=0}^{|k|-1} v_l, v_l \in k$
8: $\quad\quad$ **if** $dist(k) \leqslant \mathscr{D} \times \tau$ **then**
9: $\quad\quad\quad B \leftarrow MaxRate$
10: $\quad\quad$ **else** $B \leftarrow C_i \div Dl_i$
11: $\quad\quad$ **end if**
12: $\quad\quad$ **Test RSA restrictions**
13: $\quad$ **end for**
14: $\quad$ **if** $\exists k \in K \mid k$ **is viable then**
15: $\quad\quad \mathscr{P} \leftarrow k$
16: $\quad\quad \mathscr{B} \leftarrow B$
17: $\quad$ **end if**
18: **end for**
19: $getNext(comb)$
20: **if** $comb$ can be attended **then**
21: $\quad$ accept $(R, \mathscr{P}, \mathscr{B})$
22: **else** block (R)
23: **end if**

Definidos o caminho e a largura de banda, na linha 12 é verificada a restrição de continuidade, de contiguidade e de não-sobreposição do espectro óptico. As linhas $14 - 17$ analisam se um caminho e sua respectiva taxa foi obtida para cada requisição do subconjunto de combinações, que em caso positivo, são previamente alocadas e guardadas em $\mathscr{P}$ e $\mathscr{B}$, conjunto dos caminhos da combinação e conjunto das larguras de banda das requisições dessa mesma combinação. A linha 19 faz tentativas com todos so subconjuntos de combinações até exaurir a matriz de combinações m ou até que encontre uma combinação viável. A aceitação das MBDTs para a ressincronização é feita na linha 20, caso um subconjunto possa ser aceito. Se não for o caso, a requisição é bloqueada.

O algoritmo recebe um lote R como entrada, mas a aceitação do serviço se dá com o atendimento de apenas um subconjunto de R, o que significa que chamadas remanescentes são marcadas como bloqueadas pelo plano de controle da rede. Isso é possível porque na comunicação *cross-layer*, o roteamento, em contato com a aplicação, sabe que se R é proveniente de um grupo de replicação e portanto, os dados das transferências são similares, então algumas das requisições podem ser bloqueadas sem prejuízo do serviço.

Complexidade do Algoritmo

O RSA dinâmico com caminho fixo proposto por [Wan et al. 2012], que é dividido em dois passos, computa os *k* menores caminhos sem *loops* usando o algoritmo KSP no primeiro desses passos com tempo $O(K|V|^3)$, e em seguida, utiliza operações de detecção e intersecção de espectro no segundo passo, levando $O(K^2)$ para a verificação em todos os enlaces que compõem o caminho. Assim, seu tempo total é $O(K^3|V|^3)$.

O AA-RSA dinâmico, baseado no RSA acima, para obter as combinações de *bulks* do lote na função $COMBINATIONS(R, b)$, com um lote R de tamanho n e um fator de replicação b, tem complexidade de tempo $O\left(\frac{n!}{b!(n-b)!}\right)$ que é exponencial. Para cada elemento de uma combinação é invocada a função $KSP\,(i, K)$, com complexidade de $O(V^3)$. A atribuição de espectro é feita pela política *First-Fit* de complexidade linear. O diâmetro da rede $\mathscr{D}$ é calculado uma única vez. A partir do algoritmo de *Dijkstra*, que leva

tempo $O(V^2)$, é encontrada a distância de cada nó s do grafo para todos os demais nós desse mesmo grafo. A maior distância dentre todas as menores distâncias representa $\mathscr{D}$. Assim, a complexidade de tempo é $O\left(\left(\frac{n!}{b!(n-b)!}\right) * (V^3)\right)$. No entanto, a literatura mostra que o tamanho de R geralmente é de 3 requisições de conexões de ressincronizações, visto que no mundo real é desvantajoso, do ponto de vista do custo capital e operacional, possuir um conglomerado muito grande de recursos que são poucos solicitados [Vukolić 2010].

5. Avaliação de Desempenho

Através do simulador ONS[Costa et al.][1] desenvolvido com base no simulador WDM-Sim [Drummond 2015], simulações foram realizadas para verificar o ganho de desempenho do algoritmo proposto em relação ao algoritmo RSA convencional [Wan et al. 2012]. Eventos dinâmicos de chegadas e partidas de requisições foram simuladas na topologia NSFNET (Figura 2) com 14 nós, dos quais cinco deles $(0, 7, 11, 12, 13)$ foram definidos como CDs, e nas topologias USA e Pan-Euro reunidas por cabo de fibra óptica submarino de capacidade suficiente, com 24 e 28 nós, respectivamente (Figura 3), onde os nós $\{1, 10, 12, 20, 21, 22, 28, 30, 46\}$ são as representações de CDs. As distâncias físicas são destacadas nos enlaces. As definições dos CDs foram feitas baseadas nas localizações de centros de dados do Google [Google 2015], nas quais foram consideradas ressincronizações de uma única partição.

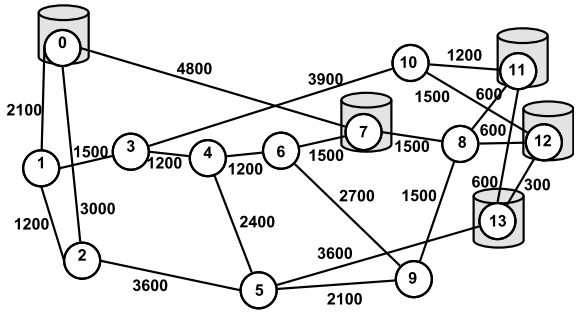


Figura 2. Topologia da rede NSFNET

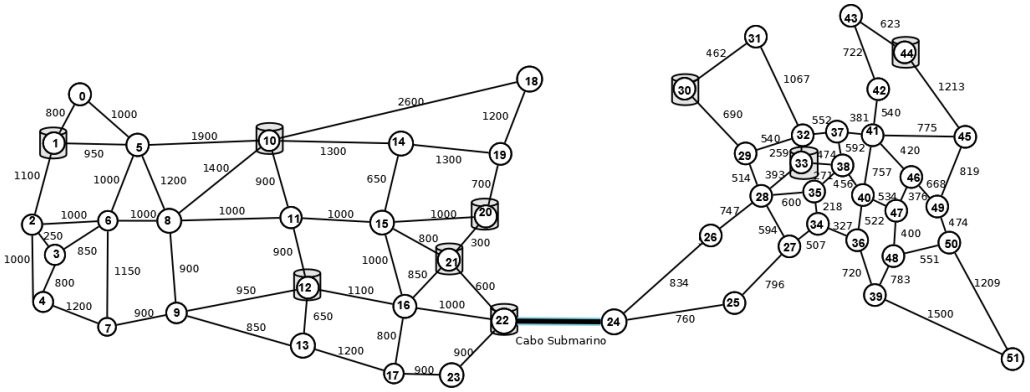


Figura 3. Topologia das redes USA e Pan-Euro, reunidas

Na implementação foram considerados 15 *transponders* por nó, cada um com capacidade de 8 *slots*. Cada *slot* possui largura de banda de $12.5GHz$ e cada enlace possui

[1]O simulador ONS é um simulador híbrido para redes WDM e EON, disponibilizado em http://comnet.unb.br/br/grupos/get/ons/download

120 *slots* de frequência. Assumiu-se ainda que são empregados dois *slots* como banda de guarda. Nos dois cenários, os dois algoritmos foram testados com as modulações *Binary Phase Shift Keying* (BPSK), que transmite um *bit* por símbolo e por isso fornece baixa taxa de dados, mostrando-se ideal para as distâncias mais longas, e *Quadrature Phase Shift Keyed* (QPSK), com dois *bits* por símbolo e uma taxa de dados mais elevada, que permite transmitir duas vezes mais quantidade de dados no mesmo canal com a mesma taxa oferecida à BPSK, e que portanto, é mais adequada para distâncias menores. Além disso, por se tratar do problema RSA, a solução não considera distâncias para definição de qualquer modulação, no entanto, o ajuste da modulação implica na definição da quantidade de *bits* por símbolo empregados no transporte dos dados. Nos testes, essas modulações tiveram resultados próximos quanto a taxa de bloqueio, embora QPSK tenha se sobressaído.

Cada simulação foi realizada 5 vezes utilizando o método de replicações independentes. Para os resultados apresentados foram calculados intervalos de confiança com 95% de confiabilidade. Foram realizadas 100.000 chamadas com origens e destinos distribuídos uniformemente dentro do subconjunto de localizações dos CDs. Os *deadlines* das chamadas foram de 10, 15 e 20 unidades de tempo. Foram empregados dois cenários de carga, um deles considerado leve, com taxas de chegadas de 2 a 10 chamadas por unidade de tempo com incrementos de 2 chegadas, e um outro cenário com tráfego pesado, cujos números de chegada de chamadas variam de 20 a 100 por unidade de tempo com incrementos de 20 chegadas. Quanto aos volumes dos *bulks*, cada terça parte das chamadas transferiram $100GB$, $500GB$ e $1TB$, respectivamente. Para essas chamadas foram configurados lotes com 3 e 4 requisições, que são parâmetros largamente encontrados na literatura [Agrawal et al. 2013b]. A simulação foi executada em uma máquina Intel Core 2Quad de 2.66GHz com 4GB de RAM, onde verificou-se que o tempo médio de execução do algoritmo AA-RSA, para atendimento de um lote, é da ordem de $1 : 27ms$ com a menor topologia e $90 : 31ms$ com a topologia reunida.

A atribuição de espectro é realizada utilizando a política *First-Fit*, onde para cada enlace pertencente a rota estabelecida, são considerados os seus respectivos BW *slots* de frequência em ordem crescente dos índices, e a demanda é acomodada iniciado-se pelos *slots* de índices menores que estão com a capacidade livre. O parâmetro τ, de restrição do diâmetro da rede utilizado no algoritmo, foi assumido como $0,5$ de modo a comparar os tamanhos dos caminhos candidatos à metade do maior caminho que pode ser encontrado.

Taxa de Bloqueio de Banda Passante (BBR)

A taxa de bloqueio de largura de banda (BBR) de *bulks* equivale a taxa do uso de largura de banda correspondente aos *bulks* bloqueados dividida pelo total de largura de banda em uso por todas as chamadas. A Figura 4 mostra que o algoritmo RSA bloqueia mais *bulks* do que o algoritmo AA-RSA no cenário das topologias reunidas, onde os caminhos ópticos estabelecidos são maiores. No cenário de tráfego leve, os algoritmos mantêm o seu comportamento até o limite de 10 chegadas. Para as duas modulações testadas, o AA-RSA apresenta melhor desempenho do que o RSA. Para ambos os algoritmos, a modulação QPSK resultou em menores taxas de bloqueio de largura de banda do que a modulação BPSK. O resultado de BBR do AA-RSA_QPSK inicia com uma taxa de cerca de 16% e chega a atingir 55% de bloqueio, ao passo que o RSA com essa mesma modulação inicializa com 28% de bloqueio, chegando a 80%.

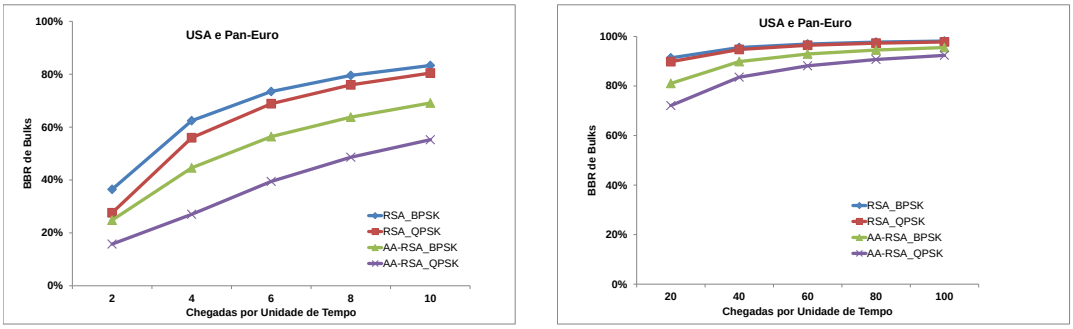


Figura 4. BBR de *bulks* nas topologias USA e Pan-Euro reunidas com tráfego leve (esquerda) e pesado (direita).

Cabe ressaltar que parte dos bloqueios do AA-RSA são na verdade chamadas descartadas dos lotes com 4 *bulks* e, portanto, não significa prejuízo no atendimento a aplicações. Quando ocorrem duas chegadas por unidade de tempo, cerca de 90% dos bloqueios dizem respeito a descartes, mas esta taxa cai para 17% quando o número de chegadas aumenta para 10. O AA-RSA_BPSK descarta menos chamadas do que o AA-RSA_QPSK. A explicação para o bom desempenho da modulação QPSK é a alta capacidade de transporte em relação a BPSK.

Para o tráfego pesado, os dois RSAs tiveram taxa de desempenho próximas e seus percentuais de utilização de BVTs e de espectro óptico foram praticamente os mesmos, embora o RSA_BPSK tenha bloqueado mais recursos mesmo após atingir o ponto de saturação a partir de 60 chegadas por unidade de tempo. As diferenças em torno de 1% entre eles, mostram que em condições pesadas de tráfego, BPSK e QPSK exibem o mesmo comportamento. Já no caso do AA-RSA, o uso da modulação BPSK levou a mais bloqueio com uma desvantagem de cerca de 4% em comparação com QPSK. O mecanismo de busca por recurso disponível desse algoritmo eleva a taxa de utilização de BVTs e de espectro óptico em pelo menos 3%, levando a uma menor BBR. Os AA-RSAs, comparados com os RSAs, aumentam a taxa de utilização desses recursos em quase 50% a medida que o número de chegadas também cresce.

Já no cenário da rede NSFNET (Figura 5), percebeu-se no gráfico à esquerda que a utilização de BVTs manteve-se em torno de 22 e 24% pelo RSA_BPSK e RSA_QPSK, respectivamente, sendo que os resultados das taxas de bloqueio de largura de banda tiveram mínimas diferenças de 1% em condições leves de tráfego. A justificativa está na quantidade e nas localizações dos nós CDs. Em um subconjunto de CDs relativamente pequeno, a possibilidade de sorteio de um nó cujos recursos nos caminhos adjacentes a ele ainda não tenha sido desocupados é muito alta. Além disso, 80% deles são concentrados no lado leste da topologia enquanto que existe um único CD situado no lado oposto, resultando em mais transferências de dados entre os membros a leste do que entre qualquer um desses membros e o CD na região oposta da rede. Os algoritmos AA-RSAs bloquearam menos recursos e novamente a modulação QPSK resultou em menor taxa de bloqueio e menor taxa de utilização de BVTs e espectro.

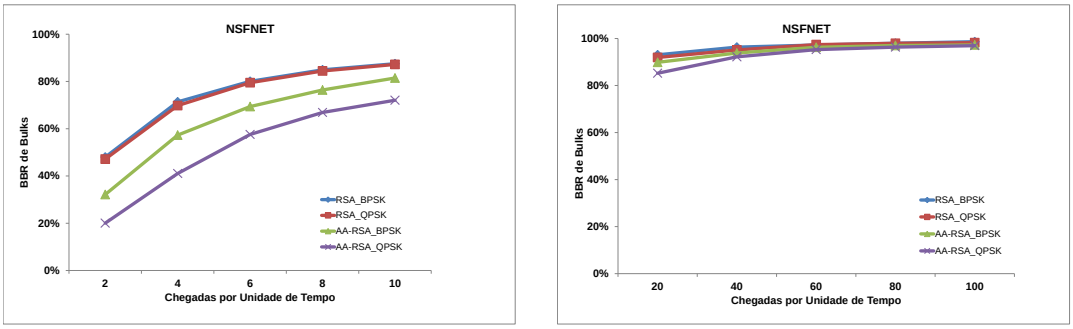


Figura 5. BBR de *bulks* na topologia NSFNET com tráfego leve (esquerda) e pesada (direita).

No gráfico à direita, percebe-se que o RSA_QPSK tem uma taxa de bloqueio por volta de 92% com 20 chegadas registradas, que aumenta para 95% com 40 chegadas e em seguida, atinge 97% na ocorrência de 60 chegadas por unidade de tempo, quando ultrapassa em quase 0,5% o RSA_BPSK, que vinha mantendo a BBR mais alta. A medida que a saturação da rede vai sendo atingida, esses dois algoritmos vão mantendo diferenças muito pequenas até que, com o número máximo de chamadas registradas o RSA_BPSK acaba bloqueado mais. Novamente, as localizações dos nós sorteados e a ocupação de recursos dos canais adjacentes com tráfego pesado justificam esses resultados. O AA-RSA_QPSK segue com menor bloqueio e experimenta uma rápida elevação logo com o aumento de tráfego, onde ocorre uma ligeira diminuição de utilização de BVTs, o que leva a entender que a taxa de ocupação da rede não permitiu o estabelecimento de novos caminhos ópticos para as requisições que solicitavam conexões. Sua diferença percentual para o AA-RSA_BPSK foi de 0,6% com a máxima taxa de chegadas registradas.

Note que nas simulações, os RSAs cientes da aplicação resultaram em menores taxas de bloqueio de largura de banda e maior utilização de BVTs devido a sua política de busca de uma combinação de chamadas para a qual existem recursos. Os RSAs clássicos tiveram taxas mais elevadas de bloqueio embora os número de recursos disponíveis como BVTs e espectro de frequência estivesse entre 6% e 12% a mais que os outros algoritmos. No geral, a modulação QPSK, de maior capacidade, demostrou melhor eficiência.

Avaliação da Qualidade do Serviço

A taxa de sucesso do serviço diz respeito às ressincronizações que se efetuaram, ou seja, o percentual de lotes atendidos. Nas topologias reunidas (Figura 6) onde se registra o tráfego leve no gráfico à esquerda, verifica-se que, como esperado, o AA-RSA_QPSK leva a um maior estabelecimento de conexões de lotes e consegue manter sua taxa de aceitação em nível elevado. O AA-RSA_BPSK tem desempenho inferior mas ainda assim, apresenta vantagens quando comparado com os RSAs clássicos. No gráfico à direita, os algoritmos cientes da aplicação tendem a manter o mesmo comportamento, mesmo com uma queda um pouco acentuada para o QPSK quando ocorrem 40 chegadas por unidade de tempo. Como a taxa de aceitação vai diminuindo, a taxa de utilização de recursos vai se reduzindo no mesmo passo. Por fim, com número máximo de chegadas, a versão com QPSK aceita 9,94% das conexões enquanto que a versão com BPSK tem taxa de 6,44%.

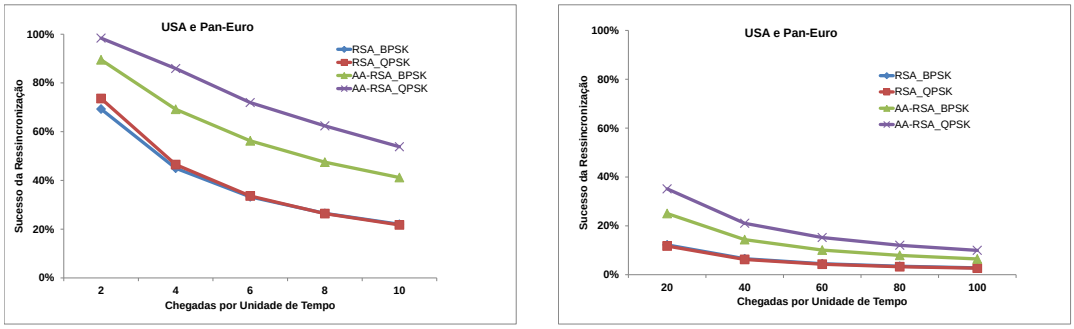


Figura 6. Taxa de sucesso de conexões atendidas nas topologias USA e Pan-Euro reunidas mediante tráfego leve (esquerda) e pesado (direita).

Quanto aos dois RSAs na Figura 6 à esquerda, como as taxas de bloqueio de largura de banda foram altas nos resultados anteriores, esse fator se reflete nas taxas de aceitação de conexões. O QPSK começa com aceitação de pouco mais de 73% enquanto que o BPSK tem 69%. Como esses dois algoritmos tratam as requisições na sequência em que são dadas, a probabilidade de atender a pelo menos 3 chamadas de um lote e resultar em uma ressincronização é pequena. O mesmo desempenho é percebido no gráfico à direita, quando seus resultados tendem a ficar muito próximos, com diferenças menores que 0,5%.

Já com relação a topologia NSFNET (Figura 7) foi registrado que a taxa de aceitação nessa rede é inferior se comparada com as topologias reunidas, mostradas anteriormente. O gráfico à direita mostra queda acentuada da taxa de aceitação do AA-RSA_QPSK em condições de tráfego pesado, em uma rede com concentração de nós. Os RSAs conservam a taxa de utilização de BVTs na faixa dos 43 a 44% e 11 a 13% de utilização do espectro, enquanto que os AA-RSAs tem variações de 28 a 32% de utilização de BVTs e até 25% de utilização do espectro. Com a diminuição de aceitações, o número de chamadas descartadas tende a diminuir enquanto que o número de chamadas efetivamente bloqueadas aumenta, e lotes acabam sendo inteiramente bloqueados devido ao alto congestionamento na rede por conexões ativas.

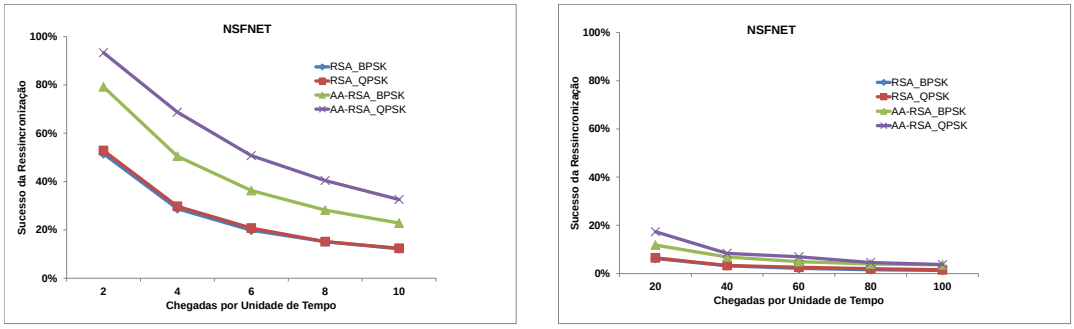


Figura 7. Taxa de sucesso de conexões atendidas na topologia NSFNET mediante tráfego leve (esquerda) e pesado (direita).

De maneira geral, os resultados mostram que algoritmos de roteamento que conhecem a aplicação para a qual precisam fornecer recursos tendem a ter melhor desempenho do ponto de vista dos serviços se comparado com algoritmos convencionais.

6. Considerações Finais

Este trabalho propõe uma solução de roteamento ciente da aplicação que eficientemente atende ao serviço de ressincronização ICD e reduz as taxas de bloqueio de banda de chamadas orientadas a dados. Os resultados comparativos mostram que a aplicação MBDT alcança maior sucesso quando a abordagem CLD é aplicada, visto que as características da aplicação podem servir de entrada para as tomadas de decisões na camada de rede.

Como trabalhos futuros pretendemos estender a proposta AA-RSA para AA-RMSA, fornecendo espaço de busca para a modulação mais adequada à distância da transmissão, adaptando as taxas oferecidas de acordo com as demandas. Com isso, será possível que mais recursos de largura de banda estejam disponíveis e assim resultem em maior aceitação de chamadas. Além disso, outros tipos de tráfegos serão acrescidos a esse cenário para simular um ambiente mais próximo da realidade, com compartilhamento dos canais de comunicação. Técnicas de escalonamento de recursos serão consideradas no atendimento das demandas com variação de tipos de tráfego e prioridades.

Referências

Agrawal, D., El Abbadi, A., Mahmoud, H., Nawab, F., and Salem, K. (2013a). Managing geo-replicated data in multi-datacenters. In Madaan, A., Kikuchi, S., and Bhalla, S., editors, *Databases in Networked Information Systems*, volume 7813 of *Lecture Notes in Computer Science*, pages 23–43. Springer Berlin Heidelberg.

Agrawal, D., El Abbadi, A., Mahmoud, H. A., Nawab, F., and Salem, K. (2013b). Managing geo-replicated data in multi-datacenters. In *Databases in Networked Information Systems*, pages 23–43. Springer.

Castro, M. and Liskov, B. (2002). Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

Christodoulopoulos, K., Tomkos, I., and Varvarigos, E. (2011). Elastic bandwidth allocation in flexible ofdm-based optical networks. *Journal of Lightwave Technology*, 29(9):1354–1366.

Costa, L. R., de Sousa, L. S., de Oliveira, F. R., da Silva, K. A., Júnior, P. J. S., and Drummond, A. C. ONS: Optical Network Simulator - WDM/EON. `http://comnet.unb.br/br/grupos/get/ons/download`.

Drummond, A. C. (2015). Wdmsim: Wdm optical network simulator. `http://www.lrc.ic.unicamp.br/wdmsim/`.

Gerber, A. and Doverspike, R. (2011). Traffic types and growth in backbone networks. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, pages 1–3.

Ghosh, A., Ha, S., Crabbe, E., and Rexford, J. (2013). Scalable multi-class traffic management in data center backbone networks. *Selected Areas in Communications, IEEE Journal on*, 31(12):2673–2684.

Google (2015). Google data centers. `https://www.google.com/maps/d/viewer?mid=zXkGqQo6GvyA.kRMGK2Zpmpbg&hl=en`. Acessed em 16/03/2015.

Jinno, M., Takara, H., Kozicki, B., Tsukishima, Y., Sone, Y., and Matsuoka, S. (2009). Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. *Communications Magazine, IEEE*, 47(11):66–73.

Laoutaris, N., Sirivianos, M., Yang, X., and Rodriguez, P. (2011). Inter-datacenter bulk transfers with netstitcher. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 74–85. ACM.

Laoutaris, N., Smaragdakis, G., Rodriguez, P., and Sundaram, R. (2009). Delay tolerant bulk data transfers on the internet. In *ACM SIGMETRICS Performance Evaluation Review*, volume 37, pages 229–238. ACM.

Li, Y., Wang, H., Zhang, P., Dong, J., and Cheng, S. (2012). D4d: Inter-datacenter bulk transfers with isp friendliness. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 597–600. IEEE.

Lu, W. and Zhu, Z. (2015). Malleable reservation based bulk-data transfer to recycle spectrum fragments in elastic optical networks. *Lightwave Technology, Journal of*, 33(10):2078–2086.

Lu, W., Zhu, Z., and Mukherjee, B. (2015). Data-oriented malleable reservation to revitalize spectrum fragments in elastic optical networks. In *Optical Fiber Communications Conference and Exhibition (OFC), 2015*, pages 1–3.

Sharov, A., Shraer, A., Merchant, A., and Stokely, M. (2015). Automatic reconfiguration of distributed storage. In *Autonomic Computing (ICAC), 2015 IEEE International Conference on*, pages 133–134. IEEE.

Song, F., Huang, D., Zhou, H., and You, I. (2012). Application-aware virtual machine placement in data centers. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 191–196. IEEE.

Subramaniam, S., Brandt-Pearce, M., Demeester, P., and Saradhi, C. V. (2013). *Cross-layer design in optical networks*. Springer.

Vukolić, M. (2010). The byzantine empire in the intercloud. *ACM SIGACT News*, 41(3):105–111.

Wan, X., Hua, N., and Zheng, X. (2012). Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *Journal of Optical Communications and Networking*, 4(8):603–613.

Zhang, H., Chen, K., Bai, W., Han, D., Tian, C., Wang, H., Guan, H., and Zhang, M. (2015). Guaranteeing deadlines for inter-datacenter transfers. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 20:1–20:14, New York, NY, USA. ACM.

Zhao, J., Wymeersch, H., and Agrell, E. (2015). Nonlinear impairment aware resource allocation in elastic optical networks. In *Optical Fiber Communication Conference*, pages M2I–1. Optical Society of America.

Comutação de Mensagens Ópticas - OMS Um Novo Paradigma para Redes Ópticas

Robson L. C. Canato[1], Gustavo B. Figueiredo[2], Juliana de Santi [3], Nelson L. S. da Fonseca[1]

[1]Instituto de Computação - Universidade Estadual de Campinas (UNICAMP)

[2]Departamento de Ciência da Computação – Universidade Federal da Bahia (UFBA)

[3]DAINF – Universidade Tecnológica Federal do Paraná (UTFPR) – Curitiba

robson@lrc.ic.unicamp.br, gustavo@dcc.ufba.br

jsanti@dainf.ct.utfpr.edu.br, nfonseca@ic.unicamp.br

Abstract. *In this work we propose a new optical switching paradigm named Optical Message Switching (OMS). The proposed approach allies the best characteristics found in its counterparts. The main contribution of our approach is to allow resource reservation in a dynamic and flexible way, aiming at promoting efficient network utilization while providing QoS guarantees. Results obtained by simulation have shown reduction in blocking probability of flows as well as reduction in the bandwidth blocking ratio, showing the benefits of the proposed paradigm.*

Resumo. *Neste trabalho, propõe-se um novo paradigma para comutação óptica em redes DWDM chamado Comutação de Mensagens Ópticas. A abordagem proposta alia o melhor encontrado na Comutação de Circuitos Ópticos (OCS), na Comutação de Rajadas Ópticas (OBS) e na Comutação de Fluxos Ópticos (OFS). A principal contribuição da abordagem proposta é permitir a reserva de recursos de forma dinâmica, visando utilização eficiente de recursos com garantia de qualidade do serviço oferecido. Os resultados demonstram redução na probabilidade de bloqueio de fluxos e na probabilidade de bloqueio de bytes, o que evidencia os benefícios do paradigma proposto.*

1. Introdução

O volume de tráfego gerado, e a consequente utilização de recursos das redes, tem aumentado devido à expansão do acesso em banda larga, ao surgimento de novas aplicações e aos novos serviços oferecidos. De acordo com o estudo apresentado em [Cisco 2014], em 2017 o número de dispositivos conectados à rede IP será quase três vezes maior que a população global e o volume de tráfego IP gerado na Internet deve atingir 1.40 zettabytes por ano. Há, assim, um crescimento exponencial no volume de tráfego nos últimos 5 anos, e a perspectiva é a manutenção desta tendência nos próximos anos [Cisco 2014]. Ademais, a diversidade de aplicações faz com que os fluxos gerados apresentem diferentes características em função de seu tamanho, duração e taxa de transmissão [Pan et al. 2012]. Tal crescimento e diversidade de tráfego requerem das redes de transporte, arquiteturas dinâmicas que sejam capazes de manter a escalabilidade exigida pela Internet, a eficiência na utilização de recursos e a qualidade dos serviços oferecidos.

Por prover grande quantidade de banda passante em seus *backbones*, as operadoras de redes de transporte tem se valido da multiplexação de vários canais ópticos operando a *40Gbps*, ou mais recentemente a *100Gbps*, oferecida pela tecnologia DWDM (*Dense Wavelength Division Multiplexing*) [Gerstel et al. 2012]. Além disso, eficiência na utilização dos recursos desta infraestrutura tem sido buscada através da eliminação de conversões óptico-elétro-óptico (O/E/O) e da proposição de novos paradigmas de comutação óptica adaptados ao novo contexto de operação das redes, o que envolve novos algoritmos e técnicas de controle sofisticados. A comutação de circuitos ópticos (*Optical Circuit Switching* - OCS) [Mukherjee 1997], a comutação de rajadas ópticas (*Optical Burst Switching* - OBS) [Qiao and Yoo 1999] e a comutação de fluxos ópticos (*Optical Flow Switching* - OFS) [Chan 2012] são paradigmas de comutação óptica largamente discutidos na literatura.

Os paradigmas de comutação óptica consideram o domínio espacial, determinando qual rota e comprimento de onda devem ser usados no aprovisionamento da demanda, e o domínio temporal, determinando em que instante de tempo a reserva dos recursos deverá ser realizada. Apesar de sua maturidade e ampla utilização em redes ópticas, a comutação OCS pode levar à baixa eficiência na utilização dos recursos disponíveis devido à flutuações no volume de tráfego transmitido através de caminhos ópticos com reserva de grande quantidade de banda passante por longos períodos de tempo. A comutação OFS é uma variação da OCS e, assim, sofre das mesma limitações da OCS. Por outro lado, a comutação OBS, que é utilizada para o encaminhamento de rajadas de tráfego pelo período de tempo necessário para a transmissão de cada rajada, não garante a reserva de recursos antes da transmissão das rajadas, o que pode levar a alta probabilidade de bloqueio e a degradação da qualidade do serviço em cenários de tráfego dinâmico [Pavon-Marino and Neri 2011]. Há, assim, a necessidade de uma forma flexível de comutação, capaz de transportar simultaneamente tráfego de diferentes características.

Neste trabalho, propõe-se um novo paradigma de comutação para redes ópticas, denominado Comutação de Mensagens Ópticas (*Optical Message Switching* - OMS), que alia as melhores características encontradas nos paradigmas OCS, OFS e OBS para utilizar os recursos de forma mais eficiente e, consequentemente, minimizar a taxa de rejeição das demandas solicitadas. Ele opera através da reserva de recursos na forma de circuitos que operam de forma não contínua considerando os domínios de tempo e espaço. Contrapondo-se aos OCS e OBS, respectivamente, o OMS provê multiplexação estatística e desacoplamento temporal entre os planos de controle e de dados.

A reserva de recursos é feita de forma dinâmica e ajustada à real demanda de tráfego, o que permite que sejam realizadas apenas pelo período de tempo em que haverá uso efetivo dos recursos, evitando a sub-utilização dos mesmos e permitindo a multiplexação estatística de tráfego com diferentes características. Além disso, leva-se ainda em consideração o compromisso entre utilização eficiente de recursos e a satisfação de requisitos de Qualidade de Serviço (QoS). Outra contribuição desse trabalho é a proposição de um algoritmo de escalonamento simples e flexível, denominado Horizon-OMS. O Horizon-OMS baseia-se no horizonte máximo e é projetado para prover a maior ocupação possível dos recursos da rede. Simulações realizadas utilizando as topologias NSF e USA mostram que o paradigma de comutação proposto é capaz de reduzir a probabilidade de bloqueio de fluxos, além de obter valores de probabilidade de bloqueio de

bytes inferiores às abordagens OCS, OFS e OBS para todos os cenários.

O restante desse trabalho é organizado da seguinte forma. A Seção 2 apresenta os trabalhos correlatos. A Seção 3 introduz a arquitetura proposta para a comutação de mensagens ópticas. Na Seção 4, apresenta-se um novo algoritmo para o escalonamento das mensagens. A Seção 5 descreve os resultados da avaliação da arquitetura proposta. Por fim, a Seção 6 apresenta as conclusões.

2. Trabalhos Relacionados

Esta seção apresenta alguns trabalhos relacionados à comutação óptica tradicional e à comutação óptica híbrida, que considera múltiplos paradigmas de comutação em um mesmo nó da rede, utilizadas como base para o desenvolvimento da arquitetura de comutação e do algoritmo de escalonamento propostos.

Em [Moura 2011] é apresentada uma avaliação do impacto produzido pela rede OBS na probabilidade de bloqueio da rede OCS. Ademais, apresenta-se uma estratégia para atualização de informações capaz de diminuir as perdas na rede devido a informações desatualizadas. Em [Perello et al. 2010], avalia-se uma rede híbrida paralela com critérios de QoS como perda e atraso de pacotes. Nessa rede, o paradigma OBS é utilizado para tráfego sensível ao atraso enquanto a rede OCS é utilizada para tráfego sensível à perdas.

Em [Vu et al. 2005], são apresentados modelos analíticos para uma rede óptica com nó híbrido integrado, considerando o caso em que os circuitos possuem prioridade sobre as rajadas e o caso em que prioridades não são consideradas. Em [Wong and Zukerman 2008], apresenta-se uma arquitetura híbrida integrada não preemptiva em que as reservas da rede OBS não são descartadas em função das reservas da rede OCS. Introduzida em [Khodashenas et al. 2011], a arquitetura de nó híbrido OBS e OCS utiliza o conceito de multi granularidade (MG-OxC), onde os comutadores englobam duas matrizes de comutação em um único nó.

Diferentemente das arquiteturas de comutação híbrida existentes na literatura, que consideram diferentes paradigmas de comutação que interagem no mesmo nó, o paradigma de comutação proposto no presente trabalho, alia somente as melhores características dos paradigmas OCS, OBS e OFS visando flexibilizar o aprovisionamento de tráfego dinâmico, utilizar a banda passante disponível de forma eficiente e garantir qualidade para os serviços fornecidos.

3. Comutação de Mensagens Ópticas

O paradigma proposto, denominado Comutação de Mensagens Ópticas (*Optical Message Switching* - OMS), capitaliza as melhores características dos paradigmas de comutação de circuitos (OCS), de comutação de fluxos (OFS) e de comutação de rajadas (OBS) para aumentar o número de fluxos aceitos, bem como utilizar os recursos disponíveis de forma eficiente e garantir a qualidade do serviço oferecido.

A motivação para o desenvolvimento do paradigma OMS leva em consideração o fato de que em redes OCS a ocupação dos circuitos é baixa quando há grandes flutuações de tráfego ou longos períodos de inatividade nas fontes de tráfego. Ademais, a garantia de serviço é provida pela reserva de recursos sempre que os mesmos são necessários,

o que não implica, necessariamente, em reserva contínua de recursos. Fica evidente, portanto, que a reserva de recursos deve ser dinâmica e flexível de forma a utilizar os recursos disponíveis com maior eficiência e, consequentemente, reduzir a probabilidade de bloqueio de demandas e melhorar a qualidade do serviço provido.

Para sanar a deficiência de dinamicidade e flexibilidade na reserva de recursos, ao invés de usar circuitos com reservas contínuas do ponto de vista temporal, a OMS propõe o uso de múltiplas reservas avançadas, através da multiplexação estatística, com menor duração, formando uma comutação híbrida. Cada reserva avançada é utilizada para transmitir uma porção dos dados, ou seja uma mensagem, do fluxo. Adicionalmente, contrapondo-se ao OBS, que provê uma relação 1:1 entre pacotes de controle e rajadas, o OMS prove uma relação de 1:N entre pacotes de controle e mensagens, significando que para um conjunto de "N" mensagens, apenas um único pacote de controle é usado.

Além disso, o número de mensagens para um fluxo é determinado em função da quantidade de dados requisitada e do tamanho máximo da mensagem em bytes, informações que são obtidas através das especificações para estabelecimento do fluxo. Assim, na abordagem proposta, utiliza-se o conjunto de reservas avançadas para comutar um conjunto de mensagens ópticas. Na comutação OMS, a duração das reservas e o tempo entre elas são projetados para coincidir com o comportamento do tráfego. Desta forma, se um dado fluxo é admitido, os recursos estarão sempre disponíveis para o tempo que são necessários. Além disso, os períodos entre reservas podem ser alocadas para comutar mensagens de outros fluxos, ou seja realizar multiplexação estatística, e, consequentemente, aumentar a capacidade de aprovisionamento de fluxos na rede. Finalmente, em um cenário de mudança de tráfego, seja por aumento da carga ou prioridade de tráfego, as últimas reservas feitas para um fluxo podem ser liberadas em favor de novas reservas, aumentando a flexibilidade para lidar com tráfego dinâmico.

3.1. Arquitetura da Rede

Na arquitetura para a rede OMS, apresentada na Figura 1, os nós de borda recebem fluxos de pacotes IP e os transmitem através dos enlaces de núcleo da rede após o processo de reserva de recursos, detalhado na Seção 3.2. Esse processo inicia-se após o nó de borda receber uma solicitação de transmissão de fluxo. A solicitação descreve o fluxo especificando o número de bytes a ser transmitido (S), e a ocupação média de um comprimento de onda ($\rho = \frac{\lambda}{u}$, sendo λ o número de bytes a serem transmitidos pelo fluxo e u a velocidade do enlace). Adicionalmente, cada comutador OMS mantém o parâmetro operacional β que indica o menor intervalo de tempo possível de duração de reserva de recursos, e é usado para determinar o número de períodos de reservas necessários para atender um determinado tamanho de fluxo.

Os pacotes recebidos a partir de camadas superiores são armazenados em filas de acordo com o nó de destino ou a especificação de QoS. Os dados são então passados para a Unidade de Adaptação de Transmissão (*Transmission Adaptation Unit* - TAU), que é responsável pela adaptação dos dados armazenados no nó de borda para a criação de um fluxo de bits de acordo com os períodos de reserva de recursos definidos para esse fluxo. Os nós de núcleo são responsáveis por escalonar os períodos de transmissão mediante a recepção de um pacote de controle, que ocorre através do canal de controle.

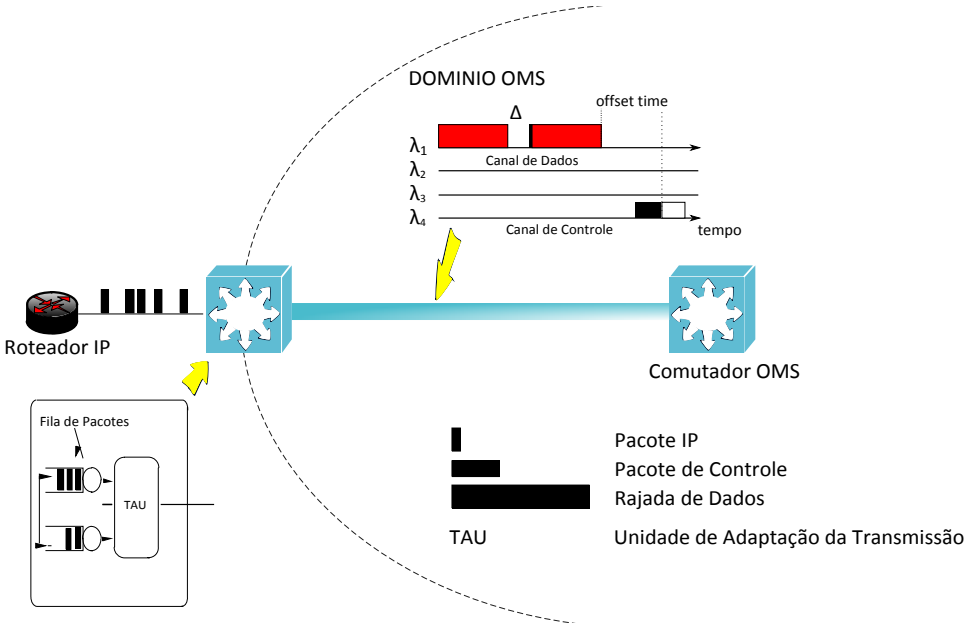


Figura 1. Esboço da arquitetura da rede OMS.

3.2. Protocolo de sinalização para reserva de recursos

A reserva de recursos em uma rede OMS é realizada pela sinalização em duas vias, o que é justificado pela necessidade de garantias ao serviço provido. A sinalização em uma única via, como a utilizada na comutação OBS, não garante que os recursos sejam reservados antes do envio dos dados, podendo gerar o descarte de dados das demandas solicitadas e, consequentemente, a degradação do serviço.

O processo de reserva de recursos na rede OMS é ilustrado na Figura 2. A descrição (S, ρ) do fluxo a ser transmitido é mapeada pelo nó de borda para:

$$info = [N, \Delta, L],$$

onde o parâmetro N denota o número de períodos de reserva (de duração β) que serão necessários para a transmissão das mensagens do fluxo; o parâmetro $\Delta = (1 - \rho) * 2\beta/\lambda$ informa o tempo mínimo entre duas reservas de um mesmo fluxo, o que é usado para combinar a intensidade do tráfego com as reservas, de modo que se ρ está próximo à taxa de transmissão do canal, Δ será zero e desse modo a reserva será um circuito contínuo. Por outro lado, se a utilização do canal é baixa, haverá um tempo equivalente a $2\beta/\lambda$ entre cada período reservado e o nível de multiplexação estatística também será incrementado; o parâmetro L lista os períodos de disponibilidade dos canais de saída do nó de extremidade.

Determinados os valores de N, Δ e L, o nó de borda envia um pacote de controle para o nó de destino para reservar recursos para o novo fluxo. Ao receber o pacote de controle, cada nó i ao longo da rota verifica a sua própria lista de disponibilidade (L) com a lista recebida pelo seu antecessor, a lista é atualizada e o pacote de controle é enviado ao próximo nó até alcançar o nó egresso.

Quando o pacote de controle é recebido pelo comutador do nó egresso, a primeira

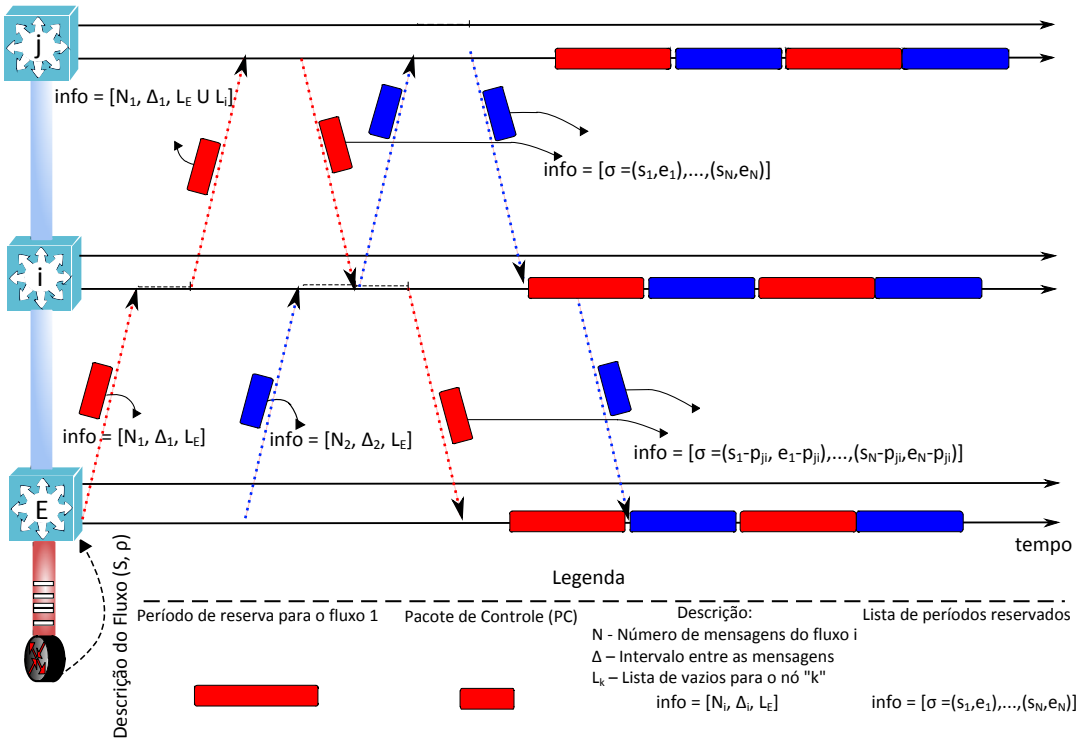


Figura 2. Processo de reserva em duas vias da rede OMS.

via é finalizada e a segunda via é inicializada. De posse da lista completa de períodos de disponibilidade nos canais ao longo da rota, o nó egresso define os períodos de tempo mais convenientes para maximizar as chances de realizar as reservas de recursos com sucesso. Em função dos períodos de tempo definidos, as reservas de recursos são escalonadas e o pacote de controle é enviado no sentido inverso da rota, na direção do nó de borda. Este segundo pacote de controle contém uma lista $\sigma = [(s_1, e_1), ..., (s_N, e_N)]$ com os instantes de tempo de início (s_i) e término (e_i) dos períodos reservados no nó egresso. Quando os nós ao longo da rota inversa recebem o pacote de controle de retorno, eles escalonam as reservas de recursos descontando o tempo de propagação da lista recebida. Se as reservas de recursos podem ser feitas em todos os nós da rota, o nó de borda recebe uma lista com os instantes de tempo em que a transmissão deverá ser feita e o processo de reserva é definido como bem sucedido.

Durante o processo de reserva, os dados recebidos de roteadores IP são armazenados no nó de borda de ingresso (Figura 1). Este fato tem consequências importantes: $i)$ não há perda de dados durante o processo de reserva; $ii)$ as escalas de tempo de operação do plano de dados e do plano de controle são desacopladas. Assim, diferentemente do que ocorre nas redes OBS, as restrições temporais sobre os nós de núcleo são menos rigorosas, permitindo que algoritmos de escalonamento mais eficientes possam ser usados sem aumentar a probabilidade de bloqueio global. Finalmente, mesmo que algumas das reservas não possam ser realizadas devido a contenção, a transmissão não precisa ser completamente bloqueada. Neste cenário hipotético, o nó de borda ingresso pode informar à rede cliente a quantidade de reservas bloqueadas e a degradação de transmissão pode ser avaliada. Se o cliente de rede estiver de acordo com a perda atingida, a transmissão pode ser realizada, o que demonstra a flexibilidade da abordagem proposta.

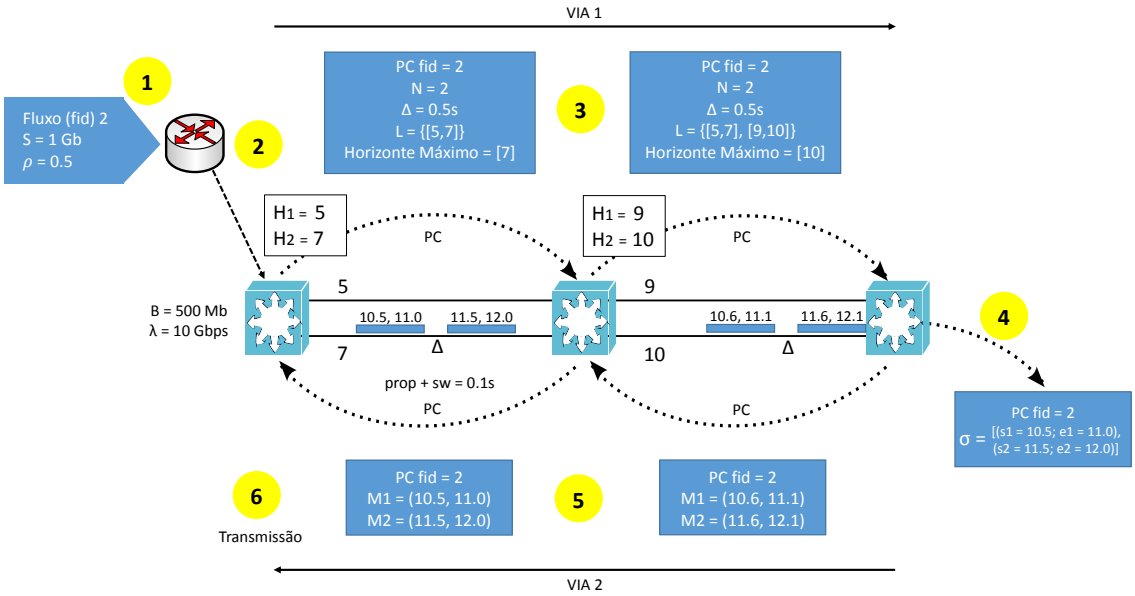


Figura 3. Processo de reserva passo-a-passo da rede OMS.

O funcionamento da arquitetura OMS é ilustrado na Figura 3. Para que um fluxo possa ser transmitido, uma solicitação é enviada para a rede (passo 1) especificando os requisitos de transmissão do fluxo ($S = 1$Gb e $\rho = 0.5$), levando-se em consideração as regras de serviço acordadas com o provedor (*Service Level Agreement* - SLA). No passo 2, um pacote de controle contendo as informações de identificação do fluxo (PC fid 2), o número de períodos de reserva ($N = 2$), o tempo mínimo entre duas reservas de um mesmo fluxo ($\Delta = 0.5s$) e a lista dos períodos de disponibilidade dos canais (L) é criado. Então, o pacote de controle é enviado ao primeiro nó da rota (passo 3). Para formar a lista de disponibilidade dos canais, o algoritmo Horizon-OMS, descrito em mais detalhes na próxima seção, verifica os horizontes além dos quais não existem mais reservas ($H_1 = 5s$ e $H_2 = 7s$) e escolhe o maior horizonte para compor a lista $L = \{[7s, ...]\}$. É importante notar que o processo de escolha do maior horizonte é realizado em cada nó ao longo da rota, de forma que ao percorrer toda a rota, a lista dos períodos de disponibilidade de canais contenha o maior horizonte de cada enlace da rota ($L = \{[7s, 10s]\}$). Quando o pacote de controle chega ao nó de egresso (passo 4), são determinados os períodos de reserva para as mensagens M_1 e M_2 e a lista de reserva $\sigma = [(s_1 = 10.5s, e_1 = 11s), (s_2 = 11.5s, e_2 = 12s)]$ com o escalonamento dos instantes de tempo de início (s_i) e término (e_i) da transmissão de cada mensagem i é criada. Cada mensagem i é escalonada no canal com maior horizonte $L = [(10s)]$ e a primeira via do processo de reserva de recursos é finalizada. A segunda via deste processo é iniciada atualizando-se o pacote de controle, com base na lista de mensagens e seus respectivos intervalos de tempo de escalonamento, que é enviado de volta na rota em direção ao nó ingresso (passo 5). Em cada nó no sentido inverso da rota, o escalonamento das reservas de recursos é realizado descontando o tempo de propagação e o tempo de comutação da lista recebida ($prop + sw = 0.1ms$), ou seja, se a mensagem M_1 iniciar sua transmissão no instante de tempo $10.5s$ no primeiro enlace da rota e o tempo de propagação e comutação for de $0.1ms$, então essa mensagem começará a ser transmitida no segundo enlace da rota no instante de tempo $10.6s$. Finalizadas as etapas de reserva de recursos, as mensagens são enviadas

à rede (passo 6) de acordo com os intervalos de tempo definidos no escalonamento das reservas de recursos $\sigma = [(10.5s, 11.0s), (11.5s, 12.0s), (10.6s, 11.10s), (11.6s, 12.1s)]$ no passo anterior.

4. Algoritmo de escalonamento de canais

O algoritmo proposto nesta seção, denominado Horizon-OMS (Algoritmo 1), é utilizado para escalonar canais em redes de comutação de mensagens ópticas. Esse algoritmo é baseado no horizonte máximo de cada comprimento de onda visto por cada nó, onde o horizonte máximo é o tempo após o qual não existe nenhuma reserva de recursos. A notação e os parâmetros utilizados no algoritmo proposto são apresentados na Tabela 1.

	Descrição
$G = (\mathcal{V}, \mathcal{E})$	é o grafo que representa a topologia da rede, onde $\mathcal{V}$ é o conjunto de nós e $\mathcal{E}$ é o conjunto de enlaces
$r^*_{s,d}$	é a rota de menor custo entre o nó de origem s e o nó de destino d
$r'_{s,d}$	representa o caminho inverso, se existir $r_{s,d} \in \mathcal{R}_{s,d}$
PC	é o pacote de controle contendo as informações do fluxo
S	é o tamanho do fluxo em bytes
B	é o tamanho máximo da mensagem em bytes
M	é a mensagem a ser enviada
k	é o número de fragmentos de mensagens que serão enviados e é dado por $k = \frac{S}{B}$
$\mathcal{H}_e$	é o conjunto de horizontes no enlace e ($e \in \mathcal{E}$)
h^*_j	é o canal de maior horizonte, dado por $h^*_j = max\{h_j, \forall j \in \mathcal{H}_e\}$
σ	é a lista de mensagens escalonadas. $\sigma = \{M_1 = (s_1, e_1), \ldots, M_k = (s_k, e_k)\}$, onde s e e são os tempos de início e término, respectivamente. σ é um atributo de PC
FB	é uma variável binária que indica o bloqueio ou não do fluxo considerado. FB é um atributo de PC

Tabela 1. Notação e Parâmetros.

O algoritmo Horizon-OMS é executado na chegada de cada requisição para estabelecimento de fluxo $f(s, d, S, B)$, onde s é o nó fonte, d é o nó destino, S é a quantidade de bytes a ser transmitida e B é o tamanho máximo da mensagem para o fluxo f. No Horizon-OMS, caso exista banda passante suficiente para aprovisonar f em cada enlace da rota entre a origem e o destino, a reserva de recursos é realizada. Caso não existam tais recursos, o fluxo é considerado como bloqueado.

O algoritmo proposto considera um grafo auxiliar G para representar as rotas existentes entre a origem e o destino do fluxo. Inicialmente, a rota de menor custo $r^*_{s,d}$ é determinada utilizando-se o tradicional algoritmo de Dijkstra (Linha 2). Então, define-se o número de fragmentos de mensagem k em função da quantidade de bytes a ser transmitida e o tamanho máximo para as mensagens do fluxo (Linha 3). Estes fragmentos são considerados na montagem da lista de mensagens escalonadas σ, que é utilizada no pacote de controle (Linha 4).

Para cada nó na rota de ida $r^*_{s,d}$ (Linha 5), caso exista um horizonte h^*_j grande o suficiente para alocar o numero de fragmentos de mensagens k (Linha 6), a lista σ é atualizada e enviada no pacote de controle para o próximo nó na rota (Linha 7). Caso contrário, na Linha 9, a rota do pacote de controle é alterada para a rota inversa $r'_{s,d}$ e a variável FB é ajustada para 1, indicando que não é possível reservar os recursos necessários para que o fluxo seja aceito.

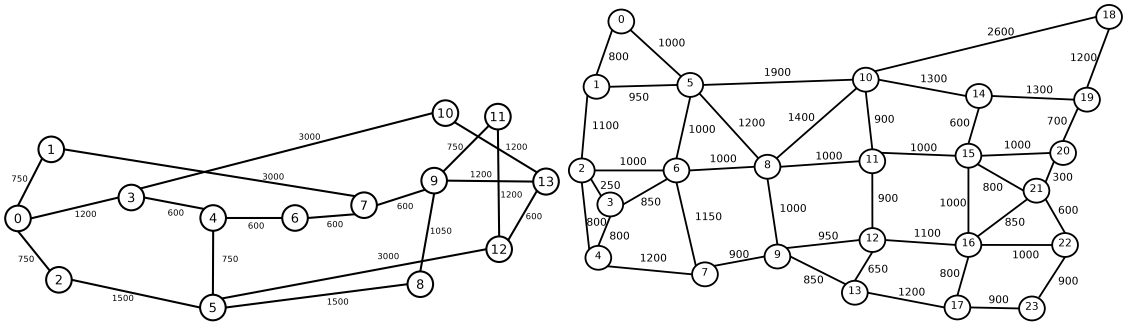
Algoritmo 1: Horizon-OMS

Entrada: G, $f(s, d, S, B)$
Saída: σ
1 $FB \leftarrow 0$
2 Use o algoritmo de Dijkstra (G) para selecionar a menor rota $r^*_{s,d}$
3 Determine o número de fragmentos de mensagem do fluxo $k \leftarrow \frac{S}{B}$
4 Crie um pacote de controle PC com σ considerando k fragmentos de M
5 **para** *cada nó na rota de ida* $r^*_{s,d}$ *até* d **faça**
6 | **se** *existir um horizonte* $h^*_j \in \mathcal{H}_e$ *grande o suficiente para alocar* k *mensagens* **então**
7 | | Atualize σ em PC e envie PC para o próximo nó
8 | **senão**
9 | | Mude a rota do PC para a rota inversa $r'_{s,d}$ e ajuste FB para 1
10 **para** *cada nó na rota inversa* $r'_{s,d}$ *até* s **faça**
11 | **se** $FB = 1$ **então**
12 | | Libere os recursos reservados listados em σ
13 | **senão**
14 | | Atualize σ em PC e envie PC para o próximo nó
15 Quando retornar ao nó s
16 **se** $FB = 0$ **então**
17 | Retorne σ
18 **senão**
19 | Retorne $\emptyset$

Terminado o processo de busca de horizontes na rota de ida, a rota inversa $r'_{s,d}$ é percorrida (Linha 10) e no caso em que não foi possível encontrar horizontes suficientes, ou seja a variável $FB = 1$ (Linha 11), os recursos previamente reservados são liberados (Linha 12). Caso os horizontes necessários em cada enlace tenham sido obtidos com sucesso, a lista σ é atualizada no pacote de controle que é enviado para o próximo nó na rota inversa (Linha 14). Quando o pacote de controle atinge o nó origem (Linha 15), se o valor da variável FB = 0, as mensagens escalonadas são transmitidas nos períodos de reserva de recursos descritos na lista σ (Linha 17). Caso contrário (Linha 19), o fluxo é considerado bloqueado.

5. Exemplos Numéricos

Para avaliar o desempenho do paradigma de comutação OMS foram realizadas simulações e comparações com os paradigmas de comutação OCS, OFS e OBS. Para este fim, desenvolveu-se um simulador utilizando a linguagem Python e sua biblioteca Simpy. O simulador recebe como entradas a requisição para estabelecimento de conexão de um comprimento de onda entre o nó de origem e o nó de destino $(s\text{-}d)$, a topologia com o custo associado a cada enlace da rota e o mecanismo de escalonamento a ser utilizado. A taxa de chegada de requisições para estabelecimento do fluxo e o tempo de duração das mesmas seguem, respectivamente, a distribuição de Poisson e a distribuição exponencial negativa. A média para o tempo de duração do fluxo é de uma unidade de tempo. As topologias utilizadas na simulações foram a topologia NSF, com 14 nós e 18 enlaces bidirecionais (Figura 4(a)) e a topologia USA, com 24 nós e 43 enlaces bidirecionais (Figura 4(b)).



(a) Topologia NSF (b) Topologia USA

Figura 4. Topologias utilizadas nas simulações.

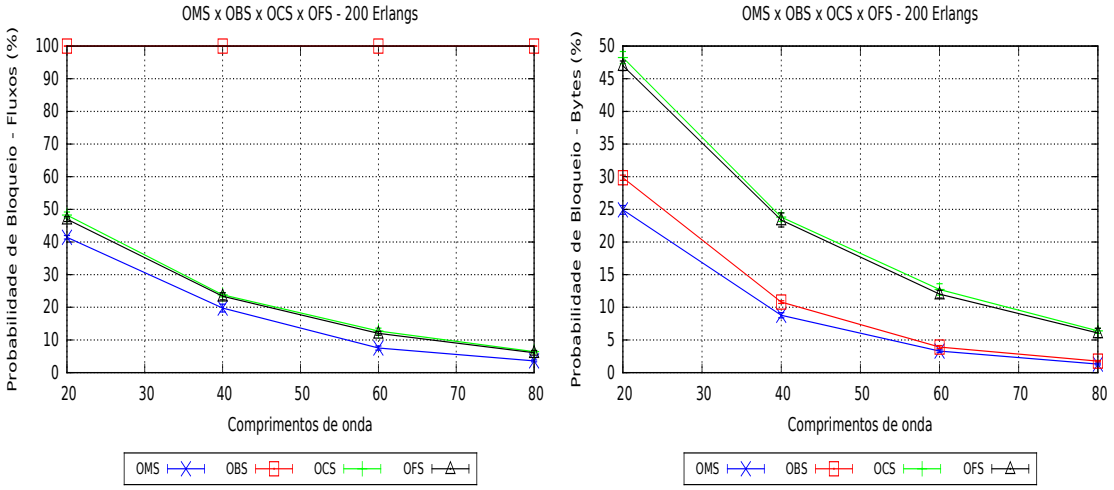
As métricas empregadas para a avaliação do paradigma proposto são a probabilidade de bloqueio de fluxos e a probabilidade de bloqueio de bytes. A probabilidade de bloqueio de fluxos é dada pela razão entre o número de fluxos bloqueados e o número total de fluxos requisitados. Um fluxo é considerado bloqueado se algum dado (byte) deste fluxo é perdido durante a transmissão. Por sua vez, a probabilidade de bloqueio de bytes é definida pela razão entre o total de bytes bloqueados e a soma dos bytes solicitados por todos os fluxos requisitados. Os bytes pertecentes a fluxos bloqueados, mas que foram transmitidos com sucesso são considerados para o total de bytes transmitidos com êxito.

Todos os gráficos apresentados nesta seção mostram valores médios obtidos a partir da execução de 10 rodadas de simulação, sendo consideradas 100.000 requisições para estabelecimento de conexão em cada rodada e utilizando-se intervalos de confiança com nível de confiança de 95%, os quais são derivados através do método da replicação independente. Esse número significativo de requisições para estabelecimento de conexão é executado de forma que o efeito transitório inicial seja desprezível, predominando o regime permanente de operação da rede.

Dois cenários são considerados nas simulações. No primeiro cenário, matém-se a carga da rede fixa em 200 Erlangs e varia-se o número de comprimentos de onda utilizando os valores 20, 40, 60 e 80 comprimentos de onda (Figura 5). No segundo cenário, mantém-se o número de comprimentos de onda fixo em 40 e varia-se a carga ofereciada à rede em 100, 200, 300, 400 e 500 Erlangs (Figura 6).

A redução na probabilidade de bloqueio de fluxo obtido pelo paradigma OMS em comparação ao paradigma OFS (Figura 5(a)) é de 11,79% e 41,19% para 20 e 80 comprimentos de onda, respectivamente. Comparado ao paradigma OCS, o OMS apresenta redução de 14,08% e 44,13% na probabilidade de bloqueio de fluxo para os mesmos comprimentos de onda. Esses resultados podem ser justificados pela reserva de recursos em múltiplos períodos, o que proporciona maior aceitação dos fluxos e, consequentemente, menor probabilidade de bloqueio. Os resultados para a probabilidade de bloqueio de bytes na topologia NSF em função do número de comprimentos de onda para carga de 200 Erlangs são apresentados na Figura 5(b). A probabilidade de bloqueio de bytes do paradigma OBS é reduzida em até 36% com 20 comprimentos de onda e até 71% quando o número de comprimentos de onda é aumentado para 80 em comparação com os paradigmas OFS e OCS. Diferentemente do OFS e OCS, como o paradigma OBS não visa prover garantia de serviço, as rajadas que pertencem à um fluxo bloqueado mas que foram

transmitidas com êxito são contabilizadas para o total de bytes transmitidos com sucesso. Assim, ao combinar os benefícios dos paradigmas OCS, OFS e OBS, a OMS é capaz de adequar-se ao tráfego dinâmico, através da otimização da duração e do intervalo de tempo entre os períodos de reserva. A utilização dos intervalos entre os períodos de reserva para realizar multiplexação estatística permite a comutação de mensagens de outros fluxos, diminuindo, dessa forma, tanto a probabilidade de bloqueio de fluxo (Figura 5(a)) quanto a probabilidade de bloqueio de bytes (Figura 5(b)). Além de melhorar a utilização dos recursos, aumenta-se a chance de atender aos requisitos de garantia de serviço, o que é cada vez mais exigido por aplicações emergentes.



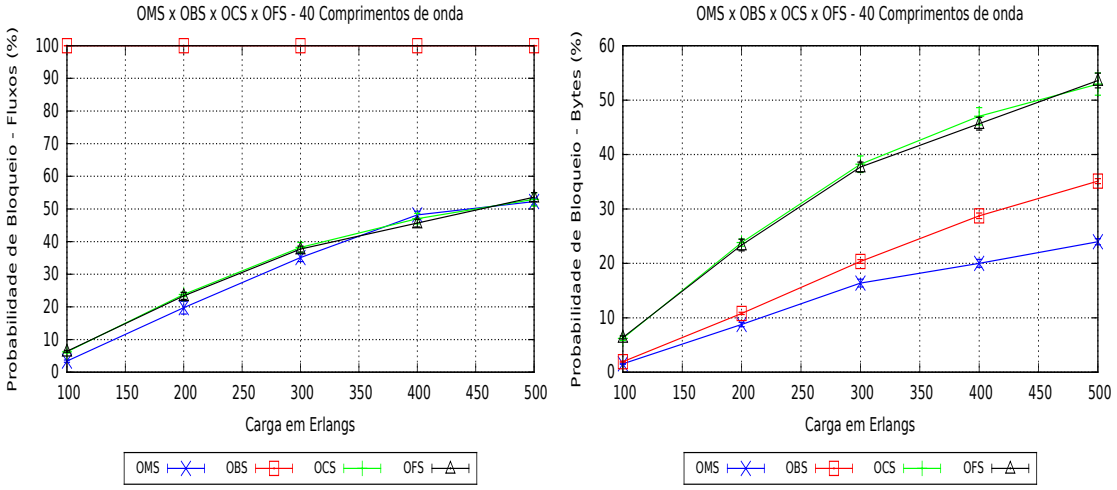
(a) Fluxos (b) Bytes

Figura 5. Bloqueio em função do número de comprimentos de onda - Topologia NSF.

As Figuras 6(a) e 6(b) apresentam, respectivamente, os resultados para a probabilidade de bloqueio de fluxos e a probabilidade de bloqueio de bytes no cenário com capacidade de 40 comprimentos de onda e variação da carga oferecida à rede.

Os valores de probabilidade de bloqueio de fluxo em função da carga para 40 comprimentos de onda na topologia NFS são apresentados na Figura 6(a). A redução da probabilidade de bloqueio de fluxo gerado pelo paradigma OMS é de 45,41% e 46,95% em comparação com os valores obtidos para a comutação OCS e OFS, respectivamente, para 100 Erlangs. Á medida que a carga na rede é aumentada, os valores de probabilidade de bloqueio produzidos pelos paradigmas OMS, OCS e OFS vão se aproximando. Esse comportamento é esperado, uma vez que os recursos tendem á se esgotar. Assim como no primeiro cenário, fica evidenciado na Figura 6(a) que os valores de probabilidade de bloqueio de fluxo gerado para o paradigma OBS é influenciado pelo descarte de qualquer rajada, levando ao bloqueio do fluxo, o que não é necessariamente crítico levando em consideração que o paradigma de comutação OBS não tem como prioridade oferecer garantia de serviço. A Figura 6(b) apresenta os valores obtidos para a probabilidade de bloqueio de bytes em função da carga para 40 comprimentos de onda na topologia NSF. A probabilidade de bloqueio de bytes gerada na OMS é reduzida em 20,10%, 75,04% e 75,74%, sob a carga de 100 Erlangs com relação aos valores gerados nos pa-

radigmas OBS, OCS e OFS, respectivamente. Mesmo com o aumento da carga para 500 Erlangs, a OMS mantém o comportamento de redução da probabilidade de bloqueio de bytes de 31,75%, 54,72% e 55,28% em comparação ao paradigma OBS, OCS e OFS, respectivamente. Esses resultados confirmam que os benefícios de garantia de reserva de recursos para múltiplos períodos e a multiplexação estatística possibilitam a utilização mais eficiente da capacidade residual da rede, contribuindo para o aumento da chance de acomodação dos bytes transmitidos, o que pode ser verificado na Figura 6(b).



(a) Fluxos (b) Bytes

Figura 6. Bloqueio em função da carga - Topologia NSF.

As Tabelas 2 e 3 apresentam os resumos de redução da probabilidade de bloqueio de fluxos e de bytes do paradigma OMS em comparação com os paradigmas OBS, OCS e OFS para as topologias NSF e USA, respectivamente. Como pode ser observado, foram considerados para essa avaliação, uma variação da carga oferecida à rede em 100, 200, 300, 400 e 500 Erlangs, utlizando os valores 20, 40, 60 e 80 comprimentos de onda. Os resultados de redução da probabilidade de bloqueio de fluxo apresentados na Tabela 2 são confirmadas em 100,00%, 70,00% e 70,00% dos valores para o paradigma OMS em relação aos paradigmas OBS, OCS e OFS respectivamente. Em apenas 30,00% desses valores a OMS teve desempenho inferior aos paradigmas OCS e OFS para as cargas de 400 e 500 Erlangs, não ultrapassando, no pior caso, 5,52%. Além disso, os valores médios de redução da probabilidade de bloqueio de fluxo apresentados para a OMS são de 70,34%, 16,99% e 16,92% em comparação ao OBS, OCS e OBS, respectivamente. Ademais, como pode-se observar na Tabela 2, o paradigma OMS reduz a probabilidade de bloqueio de bytes em todos os casos verificados na topologia NSF, sendo os valores de redução média de 26,73%, 64,43% e 64,27% em comparação à OBS, OCS e OFS, respectivamente.

Os valores obtidos de redução da probabilidade de bloqueio de fluxos e de bytes para a topologia USA são apresentados na Tabela 3. Assim como pode ser verificado na Tabela 2, a OMS apresentou na maioria dos resultados valores de redução de probabilidade de bloqueio de fluxos em relação aos paradigmas OBS, OCS e OBS. Nos resultados em que a OMS teve desempenho inferior aos paradigmas OCS e OFS esses valores não

Redução na Probabilidade de Bloqueio - %							
Carga (Erlangs)	Comprimentos de onda	OMS x OBS		OMS x OCS		OMS x OFS	
		Fluxos	Bytes	Fluxos	Bytes	Fluxos	Bytes
100	20	84,68	12,88	38,94	58,23	39,64	58,71
	40	99,61	20,10	45,41	75,04	46,95	75,74
	60	99,32	8,57	62,01	82,12	59,52	80,95
	80	99,83	50,00	63,41	90,24	71,70	92,45
200	20	58,55	16,49	14,08	48,34	11,79	46,97
	40	80,24	18,80	17,01	63,17	15,45	62,47
	60	92,49	15,86	41,01	74,16	37,47	72,61
	80	96,43	25,14	44,13	79,50	41,19	78,42
300	20	40,35	24,42	0,33	47,00	1,13	47,42
	40	64,91	19,52	8,19	57,17	6,92	56,58
	60	80,47	34,02	15,42	69,68	16,47	70,06
	80	85,62	37,50	1,98	77,16	4,52	77,76
400	20	29,70	29,23	-3,37	47,27	-5,52	46,17
	40	51,84	30,38	-2,40	57,45	-5,45	56,19
	60	67,76	34,06	5,29	67,07	6,17	67,37
	80	75,89	41,72	-5,15	74,05	-3,12	74,55
500	20	26,36	28,38	-2,26	42,84	-2,33	42,80
	40	47,75	31,75	1,30	54,72	2,52	55,28
	60	59,22	31,81	-4,24	60,33	-3,48	60,62
	80	68,82	24,02	-1,33	62,98	-3,21	62,30
Redução Média		70,34	26,73	16,99	64,43	16,92	64,27

Tabela 2. Bloqueio de fluxos e bytes - Topologia NSF.

excederam a 4,60% no pior caso. No entanto, mesmo considerando esses resultados, a redução de probabilidade de bloqueio de fluxos média obtida para a OMS em compração a OBS, OCS e OFS é de 78,67%, 21,34% e 19,04%, respectivamente. Podemos observar, ainda, na Tabela 3 que o OMS reduziu a probabilidade de bloqueio de bytes em no mínimo 95,00% dos resultados, tendo como valores médios 17,66%, 66,91% e 65,97% em comparação ao OBS, OCS e OFS, respectivamente.

Redução na Probabilidade de Bloqueio - %							
Carga (Erlangs)	Comprimentos de onda	OMS x OBS		OMS x OCS		OMS x OFS	
		Fluxos	Bytes	Fluxos	Bytes	Fluxos	Bytes
100	20	90,75	15,55	44,61	63,89	44,51	63,83
	40	98,75	0,00	58,43	78,78	56,27	77,68
	60	99,83	33,33	76,06	85,92	77,92	87,01
	80	99,88	-50,00	76,00	88,00	70,00	85,00
200	20	70,42	17,24	16,70	54,18	16,20	53,91
	40	88,32	14,21	27,09	67,60	22,34	65,49
	60	95,24	19,70	29,79	75,96	28,42	75,49
	80	97,91	5,41	34,48	78,06	29,15	76,27
300	20	56,21	23,47	8,83	51,95	8,79	51,93
	40	77,39	22,55	15,98	64,14	10,98	62,01
	60	87,66	24,05	21,50	69,47	17,95	68,09
	80	92,72	21,80	17,08	76,31	13,54	75,30
400	20	44,73	26,36	2,13	51,46	0,38	50,59
	40	64,88	24,21	-0,03	58,90	-2,63	57,83
	60	77,23	29,67	3,15	70,57	-1,20	69,24
	80	84,75	10,63	-4,60	70,03	-2,62	70,59
500	20	37,46	29,53	0,82	49,14	-2,39	47,50
	40	58,71	30,64	0,34	57,93	-1,70	57,07
	60	70,99	27,13	0,58	59,32	-2,22	58,17
	80	79,67	27,70	-2,21	66,67	-2,94	66,43
Redução Média		78,67	17,66	21,34	66,91	19,04	65,97

Tabela 3. Bloqueio de fluxos e bytes - Topologia USA.

6. Conclusões

Este artigo apresentou um novo paradigma de comutação para redes ópticas denominado comutação de mensagens ópticas - OMS. O paradigma proposto alia o melhor encontrado nos paradigmas OCS, OFS e OBS, sendo capaz de adequar-se ao tráfego dinâmico gerado por aplicações emergentes. Além disso, introduzimos o algoritmo Horizon-OMS para escalonamento das mensagens. Ao prover flexibilidade para atender novas demandas de tráfego, o paradigma OMS, utilizando o algoritmo de escalonamento Horizon-OMS, teve desempenho superior aos paradigmas OCS, OFS e OBS, reduzindo a probabilidade de bloqueio e provendo garantia de serviço. Como trabalho futuro, pretende-se desenvolver variações do algoritmo de escalonamento com diferentes requisitos.

Referências

Chan, V. (2012). Optical flow switching networks. *Proceedings of the IEEE*, 100(5):1079–1091.

Cisco (2014). The Zettabyte Era - Trends and Analysis. disponível em: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html. acessado em 2015.

Gerstel, O., Jinno, M., Lord, A., and Yoo, S. (2012). Elastic optical networking: a new dawn for the optical layer? *Communications Magazine, IEEE*, 50(2):s12–s20.

Khodashenas, P., Perello, J., Spadaro, S., Comellas, J., and Junyent, G. (2011). A feedback-based hybrid obs/ocs architecture with fast-over-slow capability. In *Optical Network Design and Modeling (ONDM), Int. Conf. on*, pages 1–6.

Moura, I., M. F. M. J. S. A. (2011). Impacto da comutação obs na probabilidade de bloqueio ocs em redes hıbridas ocs/obs. In XXX Simpósio Brasileiro de Redes de Computadores - SBRC, pages 698–711.

Mukherjee, B. (1997). *Optical Communication Networks, Ed. MacGraw-Hill*.

Pan, T., Guo, X., Zhang, C., Meng, W., and Liu, B. (2012). Alfe: A replacement policy to cache elephant flows in the presence of mice flooding. In *Communications (ICC), IEEE Int. Conf. on*, pages 2961–2965.

Pavon-Marino, P. and Neri, F. (2011). On the myths of optical burst switching. *Communications, IEEE Transactions on*, 59(9):2574–2584.

Perello, J., de Guinea, N., Spadaro, S., Junyent, G., and Comellas, J. (2010). Performance evaluation of a hybrid obs/ocs network with qos differentiation based on packet loss/delay requirements. In *Transparent Opt. Networks (ICTON), Int. Conf. on*, pages 1–4.

Qiao, C. and Yoo, M. (1999). Optical burst switching (obs) - a new paradigm for an optical internet. *Journal of High Speed Networks*, 8:69–84.

Vu, H. L., Zalesky, A., Wong, E., Rosberg, Z., Bilgrami, S., Zukerman, M., and Tucker, R. (2005). Scalable performance evaluation of a hybrid optical switch. *Lightwave Technology, Journal of*, 23(10):2961–2973.

Wong, E. and Zukerman, M. (2008). An optical hybrid switch with circuit queueing for burst clearing. *Lightwave Technology, Journal of*, 26(21):3509–3527.

Otimização de Roteamento com Protocolos Tradicionais para Aplicações Free-Viewpoint Television (FTV)

Henrique D. Garcia[1], **Priscila Solis**[1]

[1]Departamento de Ciência da Computação – Instituto de Ciências Exatas
Universidade de Brasília (UnB) – Campus Universitário Darcy Ribeiro
Caixa postal 4466 – CEP 70.910-900 – Brasília – DF – Brazil

henriquedgarcia@gmail.com, pris@cic.unb.br

Abstract. *This paper proposes a traffic engineering methodology that uses the self-similarty parameter of video traffic as a restriction on an Integer Linear Programming problem (ILP) for the definition of link costs to be used by the OSPF (Open Shortest Path First). The purpose of the methodology is to set a maximum delay value for transmission between image capture and middleboxes offering the service FTV (Free Viewpoint Television). The experimental results show that the proposed method can significantly reduce the maximum delay in the network and allow identification of threshold values for load on the links to facilitate the definition of maximum delay boundaries in video transmission environments on the Internet.*

Resumo. *Este trabalho propõe uma metodologia de engenharia de tráfego que utiliza o parâmetro de autossimilaridade do tráfego de vídeo como restrição em um problema de programação linear Inteira (PLI) para a definição dos custos dos enlaces a serem utilizados pelo OSPF (Open Shortest Path First). O objetivo da metodologia é definir um valor de atraso máximo para transmissão entre a captura de imagens e os middleboxes que oferecem o serviço de FTV (Free Viewpoint Television). Os resultados experimentais mostram que a metodologia proposta pode reduzir significativamente o atraso máximo na rede e permitir a identificação de valores de limiar para a carga nos enlaces que facilitem a definição de limites máximos de atraso em ambientes de transmissão de vídeo na Internet.*

1. Introdução

Com as demandas atuais de banda das diversas aplicações multimídia, estima-se que até 2018 mais de 80% do volume de tráfego na Internet será de vídeo [VNI 2014]. Considerando que existe um consenso sobre as caraterísticas estatísticas do tráfego multimídia serem definidas como autossimilar ou *multifractal* [Solis Barreto 2007][Sahinoglu and Tekinay 1999] [Reljin et al. 2006], torna-se um grande desafio garantir métricas de desempenho para as aplicações em uma arquitetura baseada no melhor esforço, como é o caso do TCP/IP.

Além da função básicas de transmitir dados, as redes atuais executam diversas tarefas para melhorar segurança, desempenho ou fornecer novas funcionalidades para o tráfego. Para este fim são utilizados *middleboxes*, tais como tradutores de endereços, *firewalls*, processadores de vídeo, entre outros. Quando os *middleboxes* são virtualizados,

acabam por gerar uma Rede de virtualização de funções (NFV *Network Functions Virtualization*) podendo serem inicializados ou desativados conforme necessário, aumentando a eficiência e dinamismo da rede.

Assim, os *middleboxes* têm acentuado o interesse em algoritmos que permitam:

1. Otimizar a colocação dos *middleboxes* em um ambiente de servidores de rede;
2. Acompanhar o tráfego através de sequências de *middleboxes* com base em uma política de alto nível e
3. Otimizar o roteamento do tráfego entre servidores e *middleboxes* através do cálculo de caminhos eficientes para cumprir requerimentos necessários de desempenho [Charikar et al. 2015].

Considerando que estas aplicações são hoje executadas em ambientes que utilizam principalmente o IPv4 (*Internet Protocol Verson 4*), é factível considerar a otimização do roteamento com protocolos tradicionais, tais como o OSPF, que é suportado por praticamente todos os roteadores da Internet, não exigindo modificações adicionais nos equipamentos, são estáveis e de fácil gerenciamento.

Diversos trabalhos já estudaram a otimização de roteamento com protocolos tradicionais [Papadimitriou et al. 2015] [Leduc et al. 2006] [Fortz et al. 2002] fazendo uso de métricas dinâmicas nos enlaces, porém nenhum destes trabalhos considera a autossimilaridade do tráfego como componente da métrica para definir o custo do enlace.

Altin em [Altın et al. 2013] apresenta um levantamento das pesquisas mais relevantes relacionadas com a otimização de protocolos de menor caminho. Neste escopo, Fortz em [Fortz et al. 2002] desenvolve um modelo de Programação Linear Inteira (PLI) utilizado como referência neste trabalho. Posteriormente, em [Leduc et al. 2006] implementa as heurísticas propostas por Fortz no simulador TOTEM (TOolbox for Traffic Engineering Methods) e Balon em [Balon S, Skivée F. e Leduc G., 2006] comparam de diversas funções objetivos para engenharia de tráfego, na qual o modelo de Fortz apresenta um dos melhores resultados. Por outro lado, alguns autores como em [Papadimitriou et al. 2015] buscam novas técnicas para resolver problemas de otimização relacionadas a descoberta dos custos ótimos, uma vez que a maioria destes problemas são do tipo NP-Completo.

O presente trabalho pretende abordar a otimização de roteamento do tráfego em uma rede IP tradicional com uma arquitetura baseada no uso de middleboxes para aplicações FTV (*Free-View-Point Television*), ainda em desenvolvimento e padronizaçao pela MPEG (*Moving Picture Experts Group*) [Tanimoto et al. 2011] [Scandarolli et al. 2013]. A FTV é uma aplicação que exige muito poder de processamento e alta largura de banda por transmitir e processar centenas de fluxos de vídeo ao vivo em alta resolução.

O principal objetivo é avaliar a eficiência de uma metodologia de engenharia de tráfego que considere as caraterísticas estatísticas de autossimilaridade do tráfego de vídeo agregado como restrição em um problema de programação Linear Inteira (PLI) para a descoberta dos custos dos enlaces utilizados por protocolos do menor caminho, afim de reduzir o atraso máximo dentro de sistema autônomo (AS - *Autonomous System*). Assim, é proposta a otimização do roteamento da rede afim de oferecer Qualidade de Serviço (QoS - *Quality of Service*) para o tráfego de vídeo entre a captura do vídeo e os *middleboxes* que posteriormente poderão ser disponibilizados para a comunidade.

Este artigo é organizado da seguinte forma: na seção 2 é apresentada a fundamentação teórica e revisão bibliográfica usada como base para o desenvolvimento da proposta. A seção 3 discute o modelo de otimização proposto e em seguida, a seção 4 apresenta os resultados experimentais. A seção 5 apresenta as conclusões e trabalhos futuros.

2. Fundamentação Teórica e Trabalhos Relacionados

2.1. As Aplicações de Vídeo Digital

As aplicações de vídeo consumem muito mais largura de banda do que aplicações de voz ou *web*, por exemplo . Um vídeo com resolução de 525 x 858 pixeis, 30 quadros por segundo e 16 bits de cores pode exigir uma taxa de transmissão constante de aproximadamente 216 Mbps. Logo é necessário algum mecanismo de compressão para reduzir a taxa de transmissão, principalmente nos os novos formatos de Televisão de Ultra Alta Definição (UHDTV - *Ultra High Definition* TV) que pode atingir a resolução de 7680 x 4320 pixeis.

Para transmitir vídeo pela Internet pode-se adotar uma arquitetura cliente-servidor, no qual o transmissor transmite o fluxo requisitado diretamente para o cliente ou uma arquitetura P2P(*Peer-to-Peer*), onde de forma distribuída, cada nó da rede é considerado um cliente e servidor ao mesmo tempo. Para garantir escalabilidade em redes cliente-servidor, geralmente são implementadas Redes de Distribuição de Conteúdo (CDN – *Content Delivery Network*), um sistema de servidores distribuídos pela Internet para garantir acesso à mídia de forma mais eficiente para usuários geograficamente próximos.

A *Free viewpoint TeleVision* (FTV) é uma aplicação de vídeo ao vivo que permite ao espectador total liberdade na escolha do ângulo de visão dentro da cena do vídeo um ou mais ângulos de visão de forma a trazer o ambiente para mais próximo da realidade [Scandarolli et al. 2013]

A arquitetura dessa aplicação é mostrada na Figura 1. Nesta aplicação um arranjo de câmeras captura a cena que inicialmente passa por correção de cores e ângulos das diversas câmeras. Em seguida é realizada a estimação de profundidade utilizando câmeras adjacentes para gerar um mapa de profundidade que será codificado junto com os fluxos de cada câmera antes de ser transmitido ou armazenado. No receptor, o vídeo é decodificado e os fluxos de câmeras adjacentes são interpolados junto com o mapa de profundidade, que criando um novo fluxo de vídeo em um ângulos de onde fisicamente não existem câmeras, de acordo com a solicitação do espectador. Este novo fluxo pode ser exibido por diversos tipos de dispositivos, seja 3D, estereoscópico ou 2D.

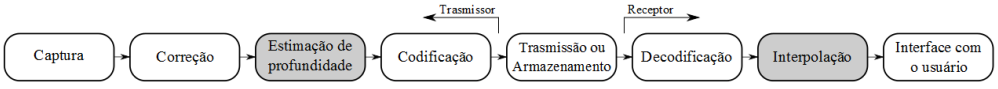


Figura 1. Arquitetura da aplicação FTV

2.2. Engenharia de Tráfego e Otimização de Roteamento

Um dos protocolos de roteamento mais antigos e sólidos na Internet é o protocolo Aberto de Primeiro Menor Caminho (OSPF - *Open Shortest Path First*) [Moy 1998], que usa como base o estado do enlace para o cálculo de rotas e funciona da seguinte maneira: após

Variável	Descrição
i, j, s, t	Nós em uma rede
N	Conjunto de todos os nós em uma rede
(i, j)	Um enlace que sai do nó i em direção ao nó j
A	Conjunto de todos os enlaces da rede
$\phi_{(i,j)}$	Custo do enlace (i,j)
(s, t)	Par origem-destino
$\delta_{(s,t)}$	Tráfego de origem em s e destino em t
$f_{(i,j)}^{(s,t)}$	Fluxo de origem s e destino t passando pelo enlace (i,j)
$l_{(i,j)}$	Carga total em um enlace
$c_{(i,j)}$	Capacidade do enlace (i,j)

Tabela 1. Notação das variáveis utilizadas para o MCNFP.

descobrir seus vizinhos, os roteadores inundam a rede com informações de seus banco de dados de rotas e custos de enlace. Assim, cada roteador constrói a topologia completa da rede e executa o algoritmo de menor caminho de Dijkstra [Bertsekas and Gallager 1992] para determinação das melhores rotas para todos os possíveis caminhos.

Na prática a descoberta dos custos ótimos é um problema de Programação Linear Inteira (PLI) do tipo NP - Completo, sendo bastante difícil encontrar uma solução ótima rapidamente. Porém, pode-se encontrar uma solução sub-ótima em tempo polinomial através de alguma heurística como *Branch and Cut*, Busca Local ou Busca Tabu [Fortz and Papadimitriou 2014].

O problema geral de roteamento, também conhecido como Problema de Roteamento de Multifluxos de Rede (MCNFP – *Multi-Commodity Network Flow Problem*) [Bertsekas and Gallager 1992] é descrito pelo PLI nas Equações 1 a 5, em que as demandas de transmissão competem entre si por recursos na rede. A notação utilizada nas equações é descrita na Tabela 1. A pesar do roteamento dentro de uma rede ser mais complexo, o MCNFP é uma boa aproximação para se encontrar os pesos de forma rápida e eficiente [Bertsekas and Gallager 1992].

$$min \quad \Phi = \sum_{(i,j)\in A} \phi_{(i,j)} \tag{1}$$

$$\sum_{j\in N} f_{(i,j)}^{(s,t)} - \sum_{j\in N} f_{(j,i)}^{(s,t)} = \begin{cases} \delta(s,t) & se\ i = s \\ -\delta(s,t) & se\ i = t \\ 0 & caso\ contrario \end{cases} \quad s,t \in N,\ (i,j) \in A \tag{2}$$

$$l_{(i,j)} = \sum_{(s,t)\in N} f_{(i,j)}^{(s,t)} \tag{3}$$

$$l_{(i,j)} \leq c_{(i,j)} \qquad (i,j) \in A \tag{4}$$

$$\phi \geq \alpha_z l_{(i,j)} - \beta_z c_{(i,j)} \quad (i,j) \in A,\ z \in Z \tag{5}$$

A Equação 1 apresenta a função objetivo que procura minimizar os custos dos enlaces da rede em função do custo ϕ. A Equação 2 é a restrição de continuidade que

garante que cada entidade que produz e consome tráfego (s, t) esteja descrita em uma matriz de tráfego δ. A Equação 3 apresenta a definição de carga como sendo a soma de todos os fluxos em um enlace (i, j), a Equação 4 garante que a carga em um enlace nunca ultrapasse sua capacidade, a Equação 5 é a função de custo que representa o aumento ou diminuição do custo dos enlaces formada por um conjunto de inequações lineares. Os pesos α e β estão associados a velocidade de crescimento do custo. No modelo proposto por este trabalho estes coeficientes são calculados realizando a regressão linear nos intervalos de interesse do modelo de atraso utilizado.

2.3. Caracterização e Modelagem do Tráfego de Vídeo em Aplicações FTV

Até meados da década de 90 o comportamento do tráfego era modelado mediante o processo de Poisson e o modelo M/M/1. Diversas pesquisas confirmaram que este modelo não era adequado para o tráfego LAN, WAN ou multimídia pois se tratavam de distribuições de cauda longa e não exponencial[Leland et al. 1994] [Feldmann et al. 1998] [Beran et al. 1995] e sua modelagem deve ser feita através de um processo que considere o índice de autossimilaridade e a sua variabilidade. Este índice de autossimilaridade também é chamado de parâmetro de Hurst (H), onde $0,5 < H < 1,0$ para séries autossimilares.

Norros em [Norros 1994], descreve um processo estocástico autossimilar chamado Processo Envelope Fractal (FEP - *Fractal Envelope Process*) caracterizado pelo valor de H, pelo desvio padrão e pela utilização da rede, ao considerar encapsular a fração $1 - \epsilon$ do tráfego gerado pelo processo do Movimento Browniano Fracionário (fBm), em que ϵ é a perda de pacotes por estouro de *buffer* dos roteadores e equivale a probabilidade do fBm ultrapassar o FEP, que será usado como o limite superior da fila. Desta forma, o número máximo de elementos em uma fila do FEP é dada pela Equação 6, em que $k = -2ln\epsilon$ e q_{max} representa o tamanho máximo da fila.

$$q_{max} = (c - \bar{a})^{\frac{H}{H-1}} (\kappa\sigma)^{\frac{1}{1-H}} (H)^{\frac{H}{1-H}} (1 - H) \tag{6}$$

Com base nessas ideias é possível estimar a largura de banda necessária para a transmissão do tráfego autossimilar e o atraso máximo da fila pelas Equações 7 e 8 respectivamente, conforme proposto em [Fonseca et al. 2000].

$$\hat{c} = \bar{a} + K^{\frac{H-1}{H}} (\kappa\sigma)^{\frac{1}{H}} H(1 - H)^{\frac{1-H}{H}} \tag{7}$$

$$t_{max} = \left(\frac{\kappa\sigma H}{c - \bar{a}} \right)^{\frac{1}{1-H}} \tag{8}$$

Quando o tráfego é agregado com vários fluxos autossimilares em um enlace, é necessário calcular o novo valor do H e da variância chamado de parâmetro H agregado ou H_g e de variância agregada σ^2 [Solis Barreto 2007], conforme mostrado nas Equações 9 e 10, respectivamente.

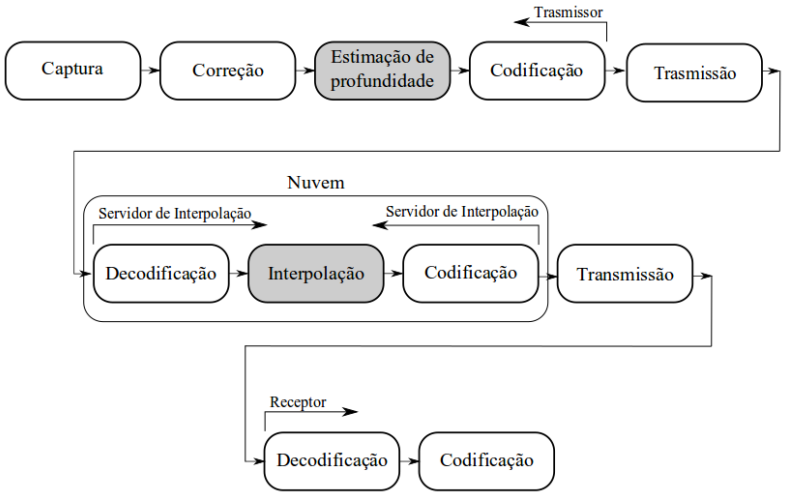


Figura 2. Arquitetura FTV utilizada neste trabalho proposta em [Scandarolli et al. 2013]

$$H_g = \frac{\sum_s H_s \sigma_s^2}{\sum_s \sigma_s^2} \tag{9}$$

$$\sigma_g^2 = \sum_s \sigma_s^2 \tag{10}$$

O tráfego de vídeo FTV consiste na transmissão de centenas de fluxos de vídeo junto com seus respectivos mapas de profundidade, tudo simultaneamente para o cliente. Como atualmente não existe um padrão de codificação para o FTV, neste trabalho foi considerado que a aplicação codifica os fluxos individualmente usando o H.265.

Para a modelagem do tráfego de vídeo utilizou-se os traços do vídeo *Tears of Steel* em resolução 4K (4096 x 1744) codificado com o padrão H.265 [Seeling and Reisslein 2014]. A taxa média de transmissão do vídeo é de 5,67 Mbps, o parâmetro H calculado é de 0,87, com um coeficiente de variação de Pearson igual $1,5$. Nesta caracterização foi considerado um fluxo agregado de 10 fluxos de vídeo, totalizando uma taxa média de transferência igual a 56,7 Mbps. Foi considerado também o $\epsilon = 0,001$ e o parâmetro H do tráfego de fundo variado de acordo com o cenário.

A arquitetura padrão do FTV não é totalmente adequada para transmissão em *broadcast* e algumas propostas tem sido feitas nesse sentido [Scandarolli et al. 2013]. A Figura 2 mostra a arquitetura proposta, onde o transmissor de vídeo poderá enviar todos os fluxos para um *middlebox* responsável pela interpolação e fisicamente próximo do cliente final através de uma arquitetura cliente-servidor, como se fosse uma Rede de Distribuição de Conteúdo (CDN - *Content Distribution Network*). Este servidor por sua vez, se comunicará com os clientes finais que solicitarão um ângulo de visão específico que será interpolado, codificado e entregue por algum sistema distribuição de fluxo de vídeo como P2P ou DASH, por exemplo. Essa arquitetura é utilizada neste trabalho para avaliar a otimização de roteamento entre a captura das imagens e o *middlebox*.

3. O modelo de otimização dos custos no OSPF

O protocolo OSPF realiza a descoberta de rotas considerando unicamente o custo estático do enlace. Considerando o crescimento da fila como um processo de Poisson, Fortz em [Fortz et al. 2002] desenvolveram um modelo linear de custos que acompanha o crescimento da fila, mas que não considera a autossimilaridade e nem a variância do tráfego.

$$\varphi' = \begin{cases} 1 & 0 \leq l/c < 2,5/100 \\ 5 & 2,5/100 \leq l/c < 5/100 \\ 20 & 5/100 \leq l/c < 8,5/100 \\ 150 & 8,5/100 \leq l/c < 15/100 \\ 1300 & 15/100 \leq l/c < 20/100 \\ 17000 & 20/100 \leq l/c < \infty \end{cases} \tag{11}$$

Com base no conceito desse modelo, neste trabalho foi desenvolvida uma função de custo descrita pela Equação 11 em que o crescimento do custo acompanha o crescimento de uma fila FEP, conforme mostrado na Figura 3. Assim, propõe-se um problema de programação linear, conforme mostrado nas Equações 12 a 22. Este modelo é chamado de LPM-FEP (*Linear Program Model - Fractal Envelope Process*)

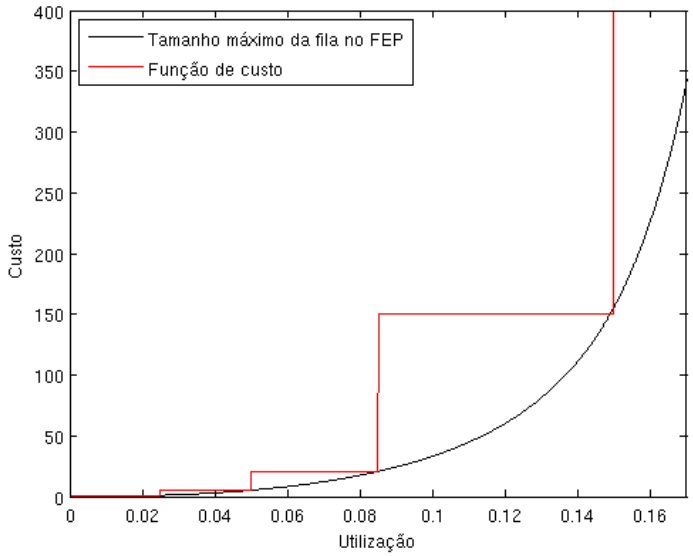


Figura 3. Crescimento do custos do enlace em função da utilização para o FEP

Para a resolução do modelo linear foi considerado como demanda a estimativa de largura de banda do FEP. Como o objetivo é atribuir custos menores aos enlaces com menor autossimilaridade agregada, e consequentemente menores atrasos máximos, o parâmetro de Hurst agregado de todos os fluxos de vídeo aparece como multiplicador do coeficiente angular, regulando o crescimento da função de custo, gerando custos maiores a enlaces com alto H_g e custos menores a enlaces com baixo H_g. O PLI foi elaborado em Uma Linguagem de Programação Linear (AMPL - *A Mathematical Programming Language*) e resolvido com o Kit de Programação Linear da GNU (GLPK - *GNU Linear Programming Kit*)

$$Minimizar\ \Phi = \sum_{(i,j)\in A} \phi_{(i,j)} \tag{12}$$

$$Sujeito\ a: \tag{13}$$

$$\sum_{j\in N} f_{(i,j)}^{(s,t)} - \sum_{j\in N} f_{(j,i)}^{(s,t)} = \begin{cases} \delta(s,t) & se\ i = s \\ -\delta(s,t) & se\ i = t \\ 0 & caso\ contrario \end{cases} \quad s,t \in N,\ (i,j) \in A \tag{14}$$

$$l_{(i,j)} = \sum_{(s,t)\in N} f_{(i,j)}^{(s,t)} \tag{15}$$

$$l_{(i,j)} \leq c_{(i,j)} \qquad (i,j) \in A \tag{16}$$

$$\phi_{(i,j)} \geq l_{(i,j)} \tag{17}$$

$$\phi_{(i,j)} \geq 5 l_{(i,j)} H_{(i,j)} - \frac{1}{10} c_{(i,j)} \tag{18}$$

$$\phi_{(i,j)} \geq 20 l_{(i,j)} H_{(i,j)} - \frac{17}{20} c_{(i,j)} \tag{19}$$

$$\phi_{(i,j)} \geq 150 l_{(i,j)} H_{(i,j)} - \frac{119}{10} c_{(i,j)} \tag{20}$$

$$\phi_{(i,j)} \geq 1300(i,j) H_{(i,j)} - \frac{922}{5} c_{(i,j)} \tag{21}$$

$$\phi_{(i,j)} \geq 17000 l_{(i,j)} H_{(i,j)} - 3325 c_{(i,j)} \tag{22}$$

A Figura 4 compara o crescimento dos custos em função da utilização entre o LPM-FEP para diversos H_g e o LPM-Fortz em escala logarítmica. Pode-se observar o crescimento do custo conforme o parâmetro de autossimilaridade aumenta. Para realizar a otimização geral da rede, baseado nos modelos propostos anteriormente, foi aplicado o *framework* mostrado na Figura 5, descrito a seguir:

- Passo 1: Caracterizar a fila dos roteadores com base o processo autossimilar da Equação 6;
- Passo 2: Obter informações da topologia e do tráfego de fundo;
- Passo 3: Calcular H_g, σ_g^2 e a banda efetiva para todos os enlaces da rede conforme Equações 9, 10 e 7 respectivamente;
- Passo 4: Gerar a matriz de tráfego com os dados gerados no passo 3;
- Passo 5: Resolver o modelo de PLI e extrair o valor dos custos dos enlaces;
- Passo 6: Inserir os novos custos na rede, obtidos no passo 5;
- Passo 7: Monitorar as métricas

Para validar a proposta, serão feitas comparações com o modelo descrito em [Fortz et al. 2002], aqui chamado de LPM-Fortz e o modelo de custo padrão da Cisco que usa o inverso da capacidade do enlace [Cisco 2015] de forma estática, ou seja $\phi_{(i,j)} = 1/c_{i,j}$, aqui chamado de Invcap.

4. Resultados Experimentais

Para simular o roteamento do tráfego foi utilizado o simulador TOTEM (*TOolbox for Trafic Engineering Methods*), que implementa diversos algoritmos para engenharia de tráfego com protocolos de menor caminho [Leduc et al. 2006].

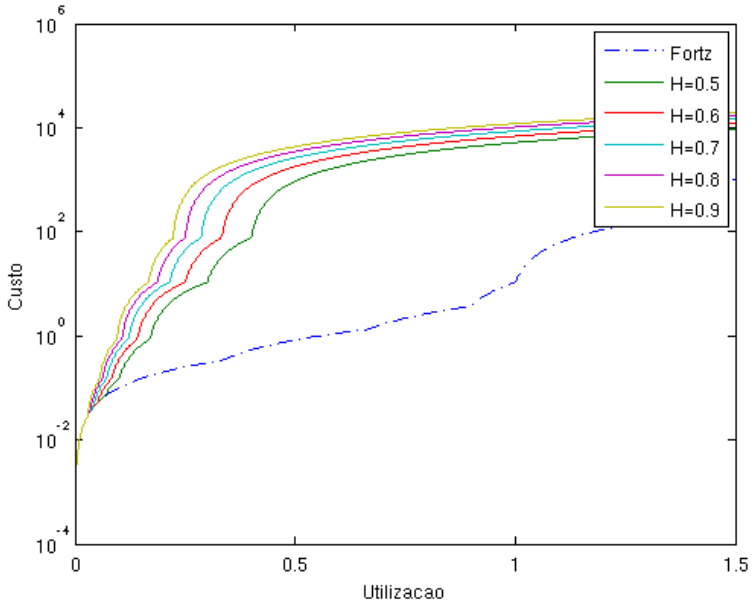


Figura 4. Comparação do crescimento do custo do enlace para o LPM-FEP com vários valores de H e o LPM-Fortz (azul pontilhado).

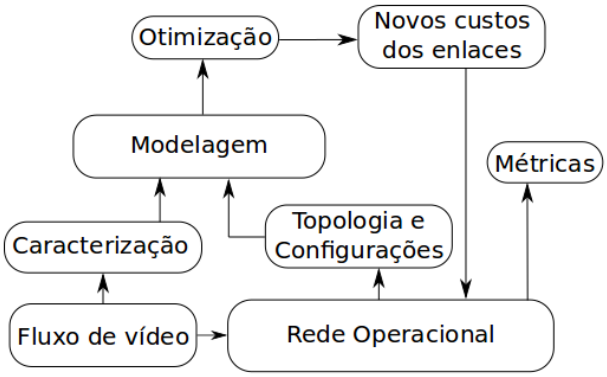


Figura 5. Fluxograma utilizado para a simulação

No TOTEM foi implementada a topologia da Rede Nacional de Pesquisa (RNP), disponível em [RNP 2015], conforme a Figura 6, porém com alteração na capacidade de alguns enlaces afim de reduzir a homogeneidade da rede. A rede conecta 27 cidades com enlaces pelo Brasil com capacidades de 100 Mbps a 20 Gbps. Foi considerado que a rede utiliza o protocolo OSPFv2 conforme definido no RFC 2328 [Moy 1998] sob IPv4, ou seja o protocolo utiliza o algoritmo de Djkstra e distribui o tráfego igualmente entre todos os enlaces que possuem o mesmo custo, técnica conhecida como Multicaminho de Mesmo Custo (ECMP - Equal-Cost Multi-Path).

Na simulação foi considerado como métrica a Média do atraso máximo da rede, que é a razão da soma dos atrasos máximos de todos os enlaces pelo número de enlaces. Esta métrica é relevante neste trabalho pois reflete o limite superior que um pacote espera no *buffer* do roteador antes de ser encaminhado para o próximo salto ou descartado até chegar ao *middlebox* que fará a distribuição aos usuários finais.

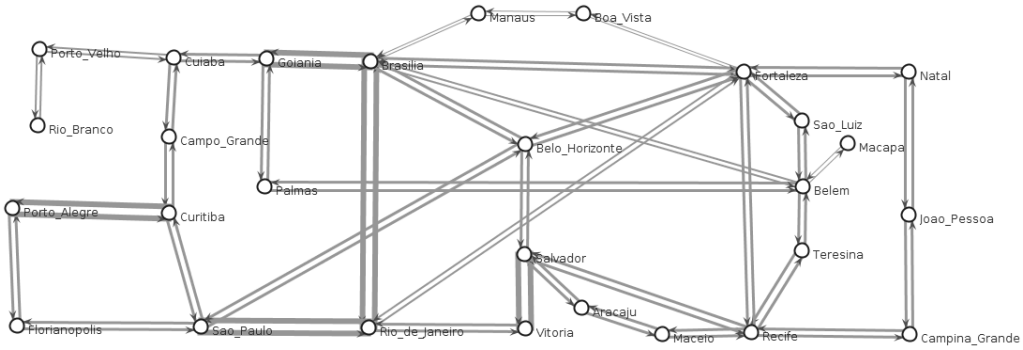


Figura 6. Topologia da Rede Ipê utilizada para simulação

De acordo com Szigeti em [Szigeti and Hattingh 2004] o atraso médio em cada nó não deve ultrapassar 5 segundos para transmissão de vídeo e em [Tanenbaum and Wetherall 2011] é especificado que até 15 segundos de atraso médio para vídeo ao vivo é aceitável. Logo, a análise dos resultados foi feita considerando esse atraso.

Para a simulação do tráfego de vídeo principal foi simulada uma demanda de tráfego autossimilar do nó Porto Alegre ao nó Belém variando de 100 Mbps a 2500 Mbps, de forma que o tráfego total sobre o enlace não ultrapasse 3000 Mbps, o coeficiente de variação igual a 1,5 e parâmetro H igual a 0,87. Todos os enlaces apresentam tráfego de fundo que varia de 5% até 20% de sua capacidade e com H aleatório entre 0,5 e 0,85 distribuído uniformemente. Foram realizadas 10 simulações para cada cenário e computados os valores médios.

A Figura 7 apresentam o atraso médio da rede em função da carga com o tráfego de fundo igual a 5% da capacidade dos enlaces. A taxa do tráfego de vídeo varia entre 100 Mbps e 2500 Mbps. O LPM-FEP manteve o atraso máximo mais baixo do que LPM-Fortz mesmo quando aplicada mais carga.

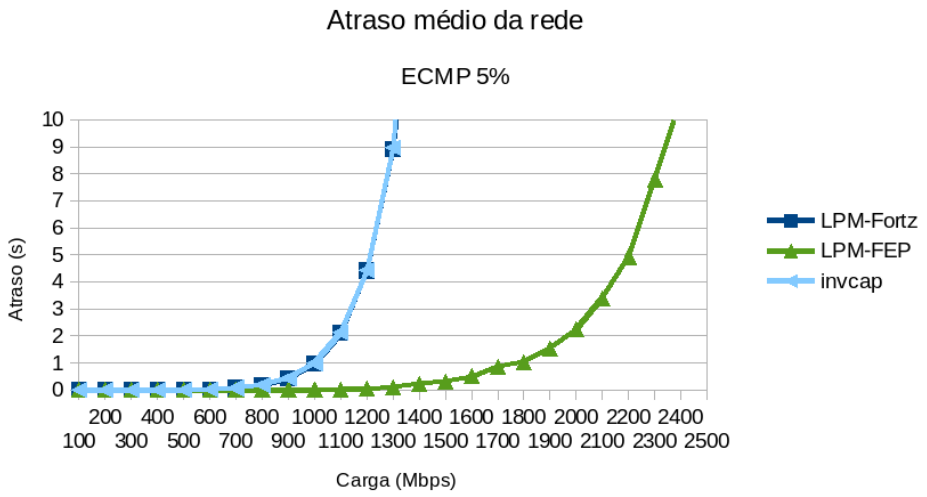


Figura 7. Atraso médio da rede com 5% de tráfego de fundo

Foram alcançados 5 segundos com 2300 Mbps de carga média na rede, 75% a

mais de carga do que LPM-Fortz e o Invcap atingiram esse mesmo atraso. Juntamente com o tráfego de fundo, esta carga corresponde a 28% da capacidade média da rede.

Quando o tráfego de fundo aumenta para 10% há um pico de atraso máximo do LPM-FEP a 200 Mbps de carga, como pode-se verificar no gráfico da Figura 8, porém logo o atraso se estabiliza e o atraso se mantém inferior aos outros modelos. Esta flutuações pode ocorrer devido ao espaçamento das utilizações na função de custo e podem ser resolvidas em um modelo de custo aperfeiçoado.

Para O LPM-FEP os valores de atraso de 5 segundos ocorrem entre 1100 Mbps e 1200 Mbps de carga, que junto como tráfego de fundo é equivalente a 22% da capacidade média da rede, enquanto o LPM-Fortz e Invcap apresentam um atraso de 5 segundos entre 900 Mbps e 1000 Mbps de carga.

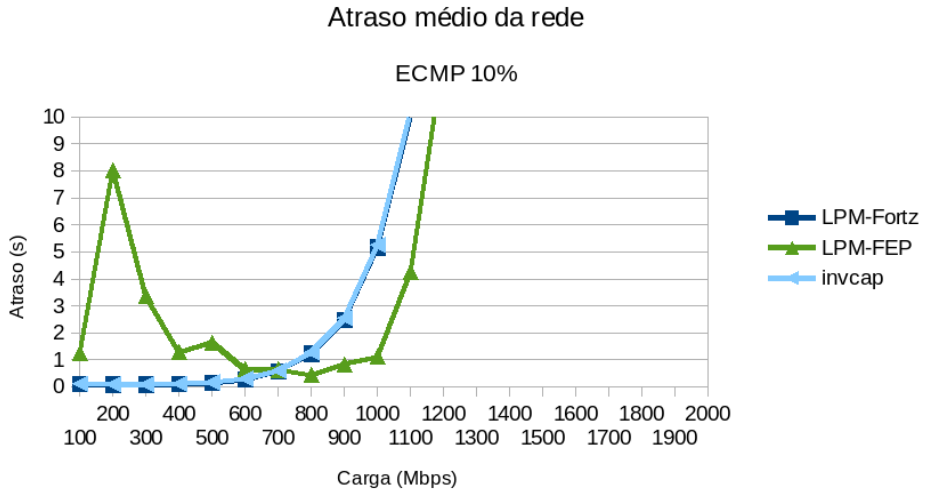


Figura 8. Atraso médio da rede com 10% de tráfego de fundo

Com 15% de tráfego de fundo, como mostra a Figura 9, o LPM-FEP apresenta uma distribuição mais uniforme do tráfego pela rede, onde pode-se observar que o atraso cresce mais lentamente, apresentando atrasos inferiores a 5 segundos em todo o intervalo de simulação, que somado ao tráfego de fundo corresponde a 30% da capacidade média da rede. Neste cenário o modelo Invcap apresenta os maiores atrasos e atinge atrasos de 5 segundo com a carga de 700 Mbps, 53% menos do que o LPM-FEP e praticamente o mesmo que o modelo LPM-Fortz.

A Figura 10 mostra os resultados para o tráfego de fundo igual a 20% a capacidade dos enlaces. O atraso médio utilizando o LPM-FEP foi superior aos LPM-Fortz até a carga de 400 Mbs e a partir deste ponto os atrasos foram inferiores a todos os outros modelos. Porém, pode-se observar que neste cenário o atraso é muito alto, atingindo mais de 10 segundos em toda a simulação, o que pode gerar atrasos fim a fim extremamente altos.

Assim, baseado no comportamento do tráfego gerado pelas simulações conclui-se que para manter o atraso médio de 99,9% do tráfego abaixo de 5 segundos por enlace, é necessário manter a carga média da rede abaixo de 25%. Este resultado baseia-se na consideração de que o tráfego de vídeo é autossimilar e que foi agregado a um tráfego de fundo que também é autossimilar.

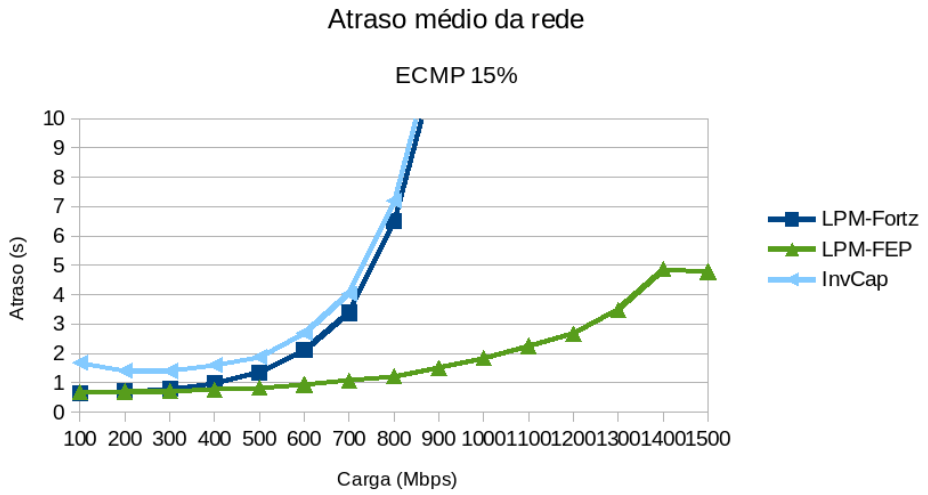


Figura 9. Atraso médio da rede com 15% de tráfego de fundo

O valor de 25% de carga média na rede pode parecer muito baixo, o que é um resultado interessante, pois essa baixa utilização é necessária para proteger a rede das rajadas de longa duração gerada pelo tráfego autossimilar. Entretanto, o Processo Envelope Fractal, estabelece valores estatísticos máximos do atraso que o tráfego alcançar. Na prática, se este modelo for avaliado em um ambiente real o atraso pode apresentar valores menores. Até a data da publicação deste artigo a utilização média da rede IPê da RNP, oscila ao redor de 16,27% [RNP 2015].

Os resultados experimentais mostram que a metodologia de otimização de roteamento proposta neste trabalho, com base um uma métrica resultante da caracterização do tráfego de vídeo autossimilar, tem o potencial de reduzir o atraso em relação ao uso de métricas tradicionais de roteamento, desde que a utilização da rede seja baixa o suficiente. Em todos os casos analisados, o atraso obtido com a proposta de otimização de roteamento deste trabalho, apresentou uma forte tendência de ser inferior do que aos outros modelos existentes.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma metodologia de engenharia de tráfego baseada na otimização do roteamento de protocolos tradicionais da pilha TCP/IP. Foi analisado e caracterizado o tráfego de vídeo digital e foi avaliado em uma arquitetura adaptada de FTV dentro de um sistema autônomo com o uso de *middleboxes*.

Com base nos parâmetros de autossimilaridade do tráfego de vídeo, foi desenvolvido um modelo de programação linear inteira para otimizar a escolha dos custos dos enlaces considerando o tráfego agregado. Foi demonstrado também que a escolha das rotas com base nesse modelo consegue reduzir o atraso de enfileiramento e manter a utilização da rede dentro de valores coerentes com outros modelos de otimização de roteamento.

Com os resultados, conclui-se de que a utilização média da rede não deve ultrapassar 25% para cumprir os requerimentos das transmissões de vídeo digital ao vivo na Internet. Este valor é um resultado interessante que demonstra a forte influencia dos pi-

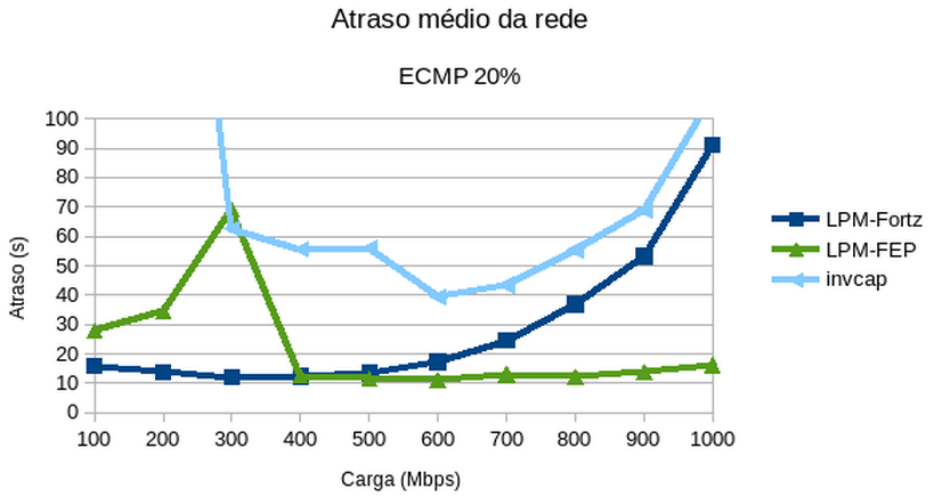


Figura 10. Atraso médio da rede com 20% de tráfego de fundo

cos de atividade de longa duração do tráfego característicos da autossimilaridade e as limitações existentes para garantir métricas de QoS para este tipo de aplicação.

Os trabalhos futuros desta pesquisa se concentrarão em aprimorar o modelo e estende-lo para outro tipo de aplicações e arquiteturas de serviço na Internet. Pretende-se aplica-lo a outros cenários e ampliar o estudos de algoritmos para resolução deste problemas de otimização.

Referências

Altın, A., Fortz, B., Thorup, M., and Ümit, H. (2013). Intra-domain traffic engineering with shortest path routing protocols. *Annals of Operations Research*, 204(1):65–95.

Beran, J., Sherman, R., Taqqu, M., and Willinger, W. (1995). Long-range dependence in variable-bit-rate video traffic. *Communications, IEEE Transactions on*, 43(2/3/4):1566–1579.

Bertsekas, D. P. and Gallager, R. G. (1992). *Data networks*. Prentice-Hall International, London. 1ère édition publiée en 1987. International student edition en 1992.

Charikar, M., Naamad, Y., Rexford, J., et al. (2015). Multi-commodity flow with in-network processing. Technical Report TR-995-15, Department of Computer Science, Princeton University, ftp://ftp.cs.princeton.edu/techreports/2015/995.pdf.

Cisco (2015). Ios quality of service solutions configuration guide. Technical report. Release 12.2, http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c.html, acessado em Set, 2015.

Feldmann, A., Gilbert, A. C., and Willinger, W. (1998). Data networks as cascades: Investigating the multifractal nature of internet wan traffic. *SIGCOMM Comput. Commun. Rev.*, 28(4):42–55.

Fonseca, N. L. S., Mayor, G. S., and Neto, C. A. V. (2000). On the equivalent bandwidth of self-similar sources. *ACM Trans. Model. Comput. Simul.*, 10(2):104–124.

Fortz, B. and Papadimitriou, D. (2014). Branch-and-cut strategies for a multi-period network design and routing problem. In *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on*, pages 128–133.

Fortz, B., Rexford, J., and Thorup, M. (2002). Traffic engineering with traditional ip routing protocols. *Communications Magazine, IEEE*, 40(10):118–124.

Leduc, G., Abrahamsson, H., Balon, S., et al. (2006). An open source traffic engineering toolbox. *Comput. Commun.*, 29(5):593–610. Disponível em `http://totem.run.montefiore.ulg.ac.be/`.

Leland, W. E., Taqqu, M. S., Willinger, W., and Wilson, D. V. (1994). On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Netw.*, 2(1):1–15.

Moy, J. (1998). Ospf version 2. RFC 2328 (INTERNET STANDARD). Updated by RFCs 5709, 6549, 6845, 6860.

Norros, I. (1994). A storage model with self-similar input. *Queueing Systems*, 16(3-4):387–396.

Papadimitriou, D., Fortz, B., and Gorgone, E. (2015). Lagrangian relaxation for the time-dependent combined network design and routing problem. In *Communications (ICC), 2015 IEEE International Conference on*, pages 6030–6036.

Reljin, I., Samčović, A., and Reljin, B. (2006). H.264/avc video compressed traces: Multifractal and fractal analysis. *EURASIP J. Appl. Signal Process.*, 2006:123–123.

RNP (2015). Topologia. na Internet. Disponível em http://www.rnp.br/servicos/conectividade/rede-ipe.

Sahinoglu, Z. and Tekinay, S. (1999). On multimedia networks: self-similar traffic and network performance. *Communications Magazine, IEEE*, 37(1):48–52.

Scandarolli, T., de Queiroz, R., and Florencio, D. (2013). Attention-weighted rate allocation in free-viewpoint television. *Signal Processing Letters, IEEE*, 20(4):359–362.

Seeling, P. and Reisslein, M. (2014). Video traffic characteristics of modern encoding standards: H.264/avc with svc and mvc extensions and h.265/hevc. *The Scientific World Journal*, 2014:16.

Solis Barreto, P. (2007). *Uma Metodologia de Engenharia de Tráfego Baseada na Abordagem Auto-Similar para a Caracterização de Parâmetros e a Otimização de Redes Multimídia*. Tese, Universidade de Brasília.

Szigeti, T. and Hattingh, C. (2004). *End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs (Networking Technology)*. Cisco Press.

Tanenbaum, A. and Wetherall, D. (2011). *Computer Networks*. Pearson, 5th edition.

Tanimoto, M., Tehrani, M., Fujii, T., and Yendo, T. (2011). Free-viewpoint TV - a review of the ultimate 3DTV and its related technologies. *Signal Processing Magazine, IEEE*, 28(1):67–76.

VNI (2014). Visual networking index - forecast and methodology, 2013–2018. Technical report, Cisco. http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html.

Salão de Ferramentas do SBRC 2016
Sessão Técnica 1
Emulação e Simulação de Cidades Inteligentes, Redes Ópticas e Redes sem Fio

Mininet-WiFi: Emulação de Redes Sem Fio Definidas por Software com suporte a Mobilidade

Ramon dos Reis Fontes[1], **Christian Esteve Rothenberg**[1]

[1]Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6101, 13083-970 – Campinas – SP – Brazil

{ramonrf,chesteve}@dca.fee.unicamp.br

Abstract. *Software-Defined Wireless Networks is being regarded as a paradigm shift in network architectures based on a clear and programmatic between control and data planes enabling the control plane innovation and the underlying infrastructure to be abstracted from applications and network services. Despite ongoing efforts to realize the vision of SDWN, many questions remain open from multiple perspectives towards their feasibility, including effective means to experiment and rapid prototype candidate applicable to the real world. To this end, we propose the emulation of wireless networks with the use of Mininet-WiFi, as a potential approach to leverage research on SDWN by using existing protocol and application stacks while mimicking the behavior of real wireless networks.*

Resumo. *As redes sem fio definidas por software (SDWN) têm sido relacionadas na mudança de paradigma em arquiteturas de redes baseado na programática entre os plano de controle e dados, permitindo a inovação no plano de controle e a utilização da infraestrutura subjacente por aplicações e serviços de rede. Apesar dos esforços na construção de uma visão sobre SDWN muitas questões ainda permanecem abertas em relação a sua viabilidade, incluindo meios efetivos para a experimentação e prototipação aplicáveis no mundo real. Sendo assim, nós propomos a emulação das redes sem fio através do Mininet-WiFi como alternativa para alavancar pesquisas em SDWN através da utilização de pilhas de protocolos e aplicações existentes ao mesmo tempo em que é capaz de reproduzir o comportamento de redes sem fio reais.*

1. Introdução

As redes sem fio consistem em um número de nós que comunicam entre si através de um canal sem fio com a vantagem da mobilidade [Holt and Huang 2010], entre outras coisas. O padrão para redes sem fio mais amplamente aceito publicado pelo Instituto Europeu de Normas de Telecomunicações (ETSI) é o IEEE 802.11, principalmente porque muitos dispositivos portáteis possuem chipsets 802.11 incorporados, além de equipamentos como pontos de acessos serem relativamente baratos.

Embora sejam muito importantes, ainda existem barreiras estruturais que impedem a inovação em redes sem fio. Além disso, as grandes infraestruturas sem fio não estão completamente acessíveis, pois existem restrições no seu uso ou exigem autenticação [Yap et al. 2010]. A questão não é abrir completa e gratuitamente o acesso

às redes sem fio, mas permitir que usuários se conectem a várias redes (preservando requisitos de segurança e de qualidade), abrindo enorme capacidade de cobertura, além de possibilitar a inovação contínua, conforme proposto em [Santos et al. 2013].

A fim de quebrar com as barreiras e caminhar rumo à inovação, propomos neste trabalho a utilização do Mininet-WiFi, emulador que permite a adoção do conceito de redes definidas por software (SDN) [Kreutz et al. 2014, McKeown et al. 2008] para redes sem fio, mais conhecido por Software Defined Wireless Networks (SDWN) [Jagadeesan 2014] [Costanzo et al. 2012] nas pesquisas e experimentações em SDWN. Através de um computador o Mininet-WiFi permite emular diferentes tipos de cenários como redes *infra-estruturadas*, *ad hoc*, *mesh*, além de permitir a mobilidade dos nós, suportando atualmente até a integração com redes veiculares. Alguns destes cenários, além de outros, estão disponíveis em um canal na Internet[1].

A emulação de rede já tem sido amplamente utilizada na avaliação de desempenho, testes e depuração de protocolos e também em pesquisas sobre redes de computadores [Fall 1999], sendo uma abordagem muito importante para o desenvolvimento de novas soluções e avaliações das atualmente existentes. Emuladores possibilitam aos usuários terem controle total sobre suas experimentações e o Mininet-WiFi, mais especificamente, permite a utilização de qualquer aplicação de rede com pouca ou nenhuma alteração no *modus operandi* dessas aplicações.

A emulação para redes sem fio é mais desafiadora, uma vez que envolve muitos fatores relacionados com o meio, como autenticação, autorização e conta, políticas, interferências e gestão da mobilidade, técnicas de seleção de canal [Schulz-Zander et al. 2014], entre outros. Alguns dos requisitos fundamentais dessas plataformas experimentais é o suporte à mobilidade e propagação do meio. Para tais fins se faz necessário utilizar modelos capazes de representar com fidelidade o movimento de terminais móveis, pessoas, etc., além do comportamento esperado na comunicação como largura de banda, atraso, latência e perda de pacotes, da mesma forma como ocorre no mundo real.

Além desta introdução, os demais assuntos tratados neste trabalho estão organizados da seguinte forma: na seção 2 contextualizamos as redes sem fio definidas por software, também discorrendo sobre os protocolos OpenFlow e CAPWAP; a seção 3 apresenta o emulador Mininet-WiFi; na seção 4 apresentamos quais serão as demonstrações a serem apresentadas no salão de ferramentas; na seção 5 são apresentadas algumas limitações e também algumas considerações para trabalhos futuros; e, finalmente, na seção 5 são apresentadas as conclusões deste trabalho.

2. Redes sem Fio Definidas por Software

As duas principais características das redes sem fio definidas por software ou SDWN (Software Defined Wireless Networks) [Jagadeesan 2014] [Costanzo et al. 2012] são o controle centralizado da rede através de software e a separação entre o plano de controle (parte inteligente da rede) e o plano de dados (responsável pelo encaminhamento de pacotes). Essa é uma abordagem já adotada pelas redes definidas por software ou SDN (Software-Defined Networking) [Kreutz et al. 2014], que permite administradores

[1]https://goo.gl/kPslqz

de rede especificarem o comportamento da rede de forma centralizada através de uma plataforma de controle, sendo o protocolo OpenFlow [McKeown et al. 2008] o mais utilizado pela comunidade acadêmica e também pela indústria.

SDWN tem sido tema de algumas pesquisas relacionadas a SDN, incluindo crescente atenção por parte das operadoras de redes móveis [Bernardos et al. 2014, Sama et al. 2015] e sua sinergia com as funções de redes virtualizadas - NFV (Network Function Virtualisation) [Han et al. 2015]. Contudo, embora a especificação 1.5 do protocolo OpenFlow esclareça como deve ser o tratamento de fluxos que entram e saiam por uma mesma porta, caso comum em redes sem fio, a atual especificação do protocolo não traz soluções para assuntos relacionados a redes wi-fi, incluindo os desafios apresentados anteriormente acerca da emulação de redes sem fio.

A separação do plano de dados e controle não é uma novidade para redes sem fio. A IETF já havia padronizado o LWAPP (*Lightweight Access Point Protocol*) - RFC541 e CAPWAP (*Control And Provisioning of Wireless Access Points*) - RFC4564 alguns anos atrás e muitas empresas utilizam esses protocolos na gestão de suas redes sem fio. O *LWAPP* é um protocolo que define as mensagens de controle para configuração, autenticação e operações em tempo real, enquanto que o *CAPWAP* é um protocolo interoperável baseado no *LWAPP* que permite que controladores administrem pontos de acesso de forma centralizada.

Existem uma série de trabalhos publicados que propõem soluções para experimentação em SDWN, a exemplo de: *OpenRoads* [Yap et al. 2010], *Odin* [Suresh et al. 2012], *OpenRF* [Kumar et al. 2013], *Ethanol* [Moura et al. 2015]. Outros trabalhos como *CloudMac* [Dely et al. 2012] e *Chandelle*[2] mencionam CAPWAP em suas soluções. Para o *CloudMac* protocolos de gerenciamento de redes sem fio como o CAPWAP são difíceis de serem estendidos e oferecer novas funcionalidades e por isso propõe uma arquitetura para a gestão de redes sem fio. O *Chandelle*, por sua vez, propõe uma solução de *roaming* para redes wi-fi de forma rápida e suave com a utilização do protocolo OpenFlow, mas também apresenta um problema na integração de comutadores tradicionais e o CAPWAP mostrando que as redes tradicionais não estão prontas para *roaming*.

Ainda hoje a única forma realística para experimentar wi-fi e OpenFlow é através da utilização de *firmware* de código-fonte aberto e soluções como o *OpenWRT*, que permite transformar simples roteadores sem fio em comutadores OpenFlow. Contudo, mesmo o *OpenWRT* possui limitações de recursos, escalabilidade, controle e reprodutibilidade, uma vez que estamos tratando com dispositivos físicos e não são todos eles que são capazes de permitir a execução da *firmware* com o protocolo OpenFlow. O Mininet-WiFi, por outro lado, é excelente quando se pretende trabalhar nestes tipos de adversidades, pois possui escalabilidade razoável, permitindo pesquisas em SDWN com a vantagem de uma rápida prototipagem e benefícios já conhecidos pela grande comunidade do Mininet [Lantz et al. 2010]. Além disso, levando em consideração que o plano de controle de redes sem fio é mais complexo que em redes cabeadas, certamente os ganhos obtidos também serão maiores.

[2]http://www.usenix.org/sites/default/files/ons2014-poster-monin.pdf

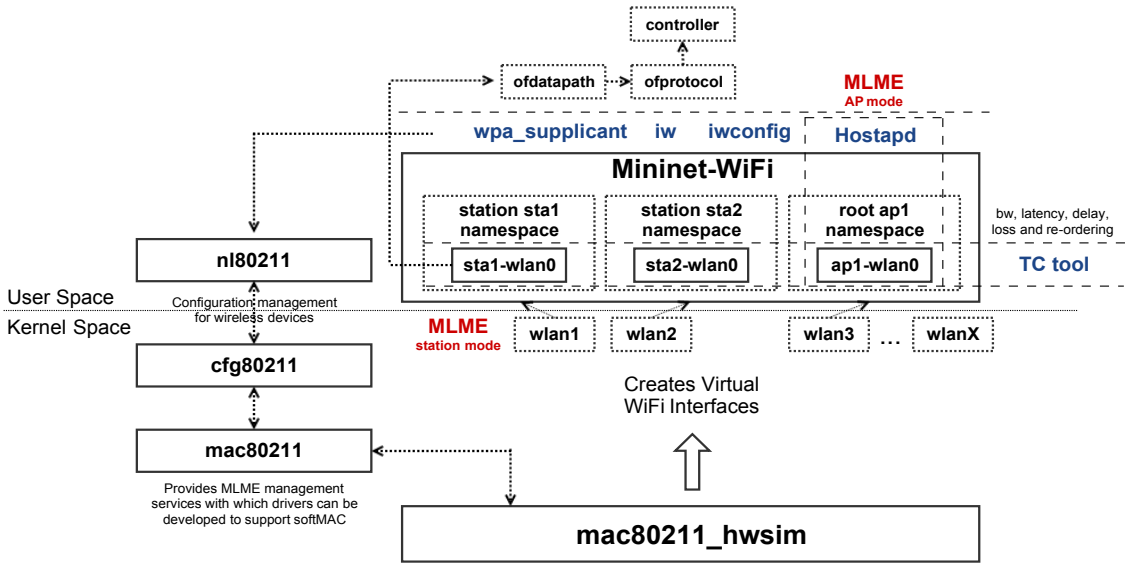


Figura 1. Componentes para funcionamento do Mininet-WiFi.

3. Mininet-WiFi

O Mininet-WiFi [Fontes et al. 2015] é um projeto de código-fonte aberto[3], desenvolvido na linguagem de programação Python e que já conta com a colaboração de vários interessados da comunidade acadêmica que estão atuando em pesquisas relacionadas a SDWN. Estações e pontos de acesso são virtualizados e os dispositivos tradicionais já presentes no Mininet [Lantz et al. 2010] também podem ser utilizados, como computadores, comutadores e controladores OpenFlow. Uma vez permitindo a utilização de controladores OpenFlow, o Mininet-WiFi possibilita o tratamento de fluxos de pacotes utilizando o protocolo OpenFlow [McKeown et al. 2008] da mesma forma como já é realizado no Mininet para experimentos em redes definidas por software (SDN) [Yang et al. 2014].

Baseado no Mininet, o principal emulador para redes OpenFlow e SDN e também no *driver* wi-fi mais utilizado em sistemas Linux, o *SoftMac*, o principal objetivo do Mininet-WiFi é explorar as características das redes wi-fi e colaborar com o avanço em *Software-Defined Wireless Networking*. Os principais componentes envolvidos no funcionamento deste emulador estão apresentados na Figura 1. No espaço do Kernel, o módulo *mac80211_hwsim* é responsável por criar as interfaces wi-fi virtuais necessárias para a criação das estações e pontos de acessos. Ainda no espaço do Kernel o MLME[4] (*Media Access Control Sublayer Management Entity*) é realizado nas estações, enquanto no espaço de usuário o *Hostapd* é o responsável por realizar essa tarefa. O *Hostapd* é o software que faz com que uma placa de rede wi-fi virtual comporte-se como uma interface de ponto de acesso.

No Mininet-WiFi também são utilizados os utilitários *iw*, *iwconfig* e o *wpa_supplicant*. Os dois primeiros são utilizados para suporte na configuração e obtenção de informações das interfaces sem fio e o último é utilizado juntamente com o *Hostapd*,

[3]`https://github.com/intrig-unicamp/mininet-WiFi`

[4]Algumas das funções realizadas pelo MLME são a autenticação, associação, envio e recebimento de *beacons*, etc.

dentre outras coisas, para o suporte ao WPA (*Wi-Fi Protected Access*). Além deles, uma outra ferramenta fundamental para o funcionamento do Mininet-WiFi é o *TC* (*Traffic Control*). O *TC* é responsável por fazer o controle das interfaces virtuais, fazendo com que diferentes valores para largura de banda, atraso, latência e perda de pacotes sejam atribuídos a elas, permitindo representar com maior fidelidade o comportamento de troca de dados do mundo real.

Para instalar o Mininet-WiFi basta seguir as 4 (quatro) etapas que estão descritas na página do código-fonte. O principal requisito para instalar e executa-lo é o sistema operacional Linux, em especial distribuições Ubuntu a partir da versão 14.04. É possível que ele funcione em outros sistemas operacionais Linux, contudo, ainda não foram realizados testes nesse sentido. Também não há uma documentação particular ao Mininet-WiFi, porém, como ele é baseado no Mininet ele incorpora toda a arquitetura do Mininet, acrescido dos recursos wi-fi apresentados na Figura 1. Além disso, os usuários contam com uma comunidade ativa[5] onde muitas questões podem ser discutidas e esclarecidas.

Uma introdução ao emulador Mininet-WiFi foi previamente publicada em [Fontes et al. 2015], no entanto, o emulador está em contínuo desenvolvimento e novos recursos estão sendo disponibilizados à medida que são implementados. Alguns desses recursos são: (i) a mobilidade, onde têm sido aprimorada nas novas versões através da implementação de novos modelos de mobilidade; (ii) propagação do sinal, onde modelos de propagação conceituados têm sido implementados, como o *Friis Propagation Loss Model* (também conhecido como *Free Space Path Loss*) e o *Two-Ray Ground Propagation Loss Model* [Rappaport 2001]; (iii) roteamento para redes mesh, com a possibilidade de utilizar um roteamento personalizado; (iv) integração com redes veiculares, onde já existe suporte para um simulador veicular bastante conhecido pela comunidade acadêmica, o SUMO[6]; entre outros.

4. Demonstração

Para demonstrar o funcionamento do Mininet-WiFi, apresentaremos quatro casos de uso destacando as evoluções desde a publicação anterior. O primeiro caso diz respeito a criação de uma simples topologia com duas estações e um ponto de acesso. No segundo caso utilizaremos uma topologia com pontos de acesso, comutadores, estações, computadores e controlador OpenFlow. No terceiro caso, mostraremos um cenário de mobilidade, onde é possível, dentre outras coisas, verificar o conceito de *handover*. E no último e quarto caso, permitiremos que usuários presentes na demonstração possam se conectar através de seus dispositivos móveis à uma rede emulada. Alguns comandos internos recentemente desenvolvidos e apresentados na página wiki[7] também poderão ser utilizados durante a demonstração, além do recurso gráfico oferecido pelo emulador (Figura 2).

No **caso 1**, será demonstrado como é realizada a comunicação entre duas estações através do comando `ping`, além de verificar a conectividade entre estações e pontos de acesso através do comando `iwconfig`. No **caso 2**, mostraremos que a comunicação entre estações e computadores é realizada com êxito quando um controlador interno em modo reativo é utilizado. Em seguida definiremos um controlador externo inexistente

[5]https://goo.gl/rRQX42

[6]http://sumo.dlr.de/wiki/Main_Page

[7]https://github.com/intrig-unicamp/mininet-wifi/wiki

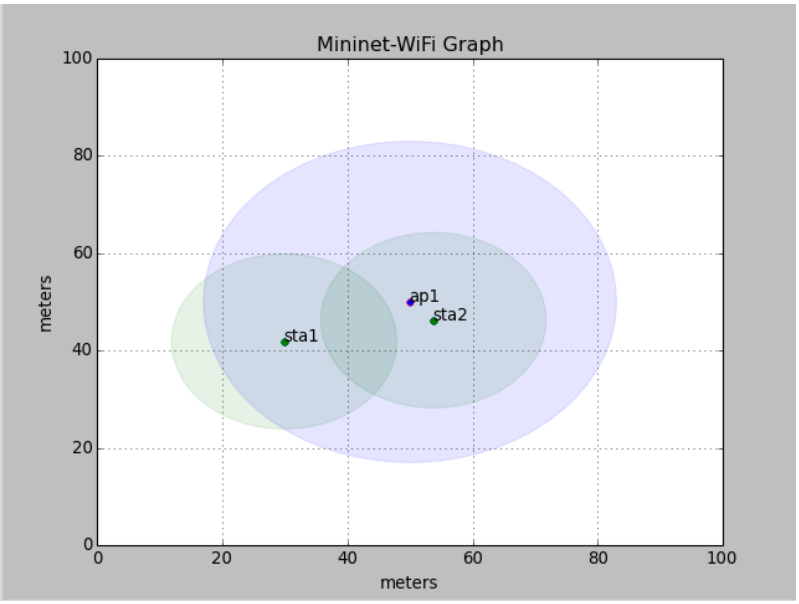


Figura 2. Mininet-WiFi em execução.

com o intuito de mostrar a importância que o controlador OpenFlow têm na comunicação entre os componentes de rede. No **caso 3**, além de verificar o conceito de *handover*, também mostraremos o impacto que a distância pode causar na comunicação entre duas estações em movimento. E finalmente, no **caso 4**, iremos criar uma topologia em ambiente emulado e qualquer usuário presente na demonstração poderá utilizar seus equipamentos móveis de forma a interagir com o ambiente emulado. Além destes casos, uma série de outros estão disponíveis em um canal[8] na Internet e no diretório *examples* (localizado no diretório raiz de instalação do Mininet-WiFi).

5. Limitações e Trabalhos Futuros

Algumas limitações têm sido percebidas durante o desenvolvimento do Mininet-WiFi, a exemplo do módulo responsável pela criação das placas de redes virtuais, o *mac80211_hwsim*, que limita a 100 (cem) o número de placas de redes que podem ser criadas numa única topologia. Por outro lado, é possível recompilar o módulo apenas alterando uma limitação imposta pela função `init_mac80211_hwsim`. Não há nenhuma referência que justifique esse limite e por isso não sabemos explicar o motivo desta limitação. Em testes realizados com o módulo recompilado conseguimos criar placas virtuais superior ao limite imposto pela função sem demais problemas.

Com relação à propagação no meio ainda existem limitações que devem ser consideradas como, por exemplo, responder com maior exatidão as interferências de sinal e taxa de transferência de dados coerente com o encontrado em um ambiente real. Estas limitações ainda serão tratadas e por esse motivo também são consideradas como trabalhos futuros. A solução para elas deve começar pela utilização de modelos de propagação já consagrados (além dos atualmente suportados) e bastante conhecidos pela comunidade acadêmica, como o *Log Normal Shadowing* [Rappaport 2001], *Nakagami* [Nakagami 1960], *Okumara-hara* [Hata 1980], dentre outros.

[8]https://goo.gl/kPslqz

6. Conclusões

Neste trabalho apresentamos a ferramenta Mininet-WiFi, um emulador para redes sem fio definidas por software (SDWN). Devido ao sucesso e sua ampla adoção pela comunidade de pesquisa, o protocolo OpenFlow é apresentado como alternativa para SDWN. Por não haver ainda definições sobre o seu funcionamento em redes sem fio, acreditamos que o emulador Mininet-WiFi pode trazer uma série de ganhos, sendo objeto de suporte em pesquisas sobre SDWN com a utilização do protocolo OpenFlow, CAPWAP e outros protocolos que venham a surgir nesse segmento, por exemplo no contexto de virtualização de funções de rede (NFV - Network Function Virtualization).

Como foi visto, embora a separação do plano de dados e controle não seja uma novidade em redes sem fio, ainda não há um definição clara sobre como deve ser o seu funcionamento, faltando plataformas de referência que forneçam suporte para o desenvolvimento de novas pesquisas. Sendo assim, propomos a utilização do emulador Mininet-WiFi, onde demonstramos alguns dos recursos atualmente suportados através de alguns casos de uso, destacando as evoluções que o emulador teve desde a sua publicação anterior. Discutimos também algumas limitações da implementação atual e considerações de resolução dessas limitações em futuras versões do emulador.

Agradecimentos

Trabalho parcialmente financiado pelo processo nº 2014/18482-4, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências

Bernardos, C., La Oliva, A., Serrano, P., Banchs, A., Contreras, L. M., Jin, H., and Zuniga, J. C. (2014). An architecture for software defined wireless networking. *Wireless Communications, IEEE*, 21(3):52–61.

Costanzo, S., Galluccio, L., Morabito, G., and Palazzo, S. (2012). Software Defined Wireless Networks: Unbridling SDNs. In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pages 1–6. IEEE.

Dely, P., Vestin, J., Kassler, A., Bayer, N., Einsiedler, H., and Peylo, C. (2012). CloudMAC - An OpenFlow based architecture for 802.11 MAC layer processing in the cloud. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 186–191. IEEE.

Fall, K. (1999). Network emulation in the vint/ns simulator. In *Proceedings of the fourth IEEE Symposium on Computers and Communications*, pages 244–250.

Fontes, R., Afzal, S., Brito, S., Santos, M., and Esteve Rothenberg, C. (2015). Mininet-WiFi: emulating Software-Defined wireless networks. In *2nd International Workshop on Management of SDN and NFV Systems, 2015*, Barcelona, Spain.

Han, B., Gopalakrishnan, V., Ji, L., and Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. *Communications Magazine, IEEE*, 53(2):90–97.

Hata, M. (1980). Empirical formula for propagation loss in land mobile radio services. *Vehicular Technology, IEEE Transactions on*, 29(3):317–325.

Holt, A. and Huang, C.-Y. (2010). *802.11 Wireless Networks: Security and Analysis*. Springer Publishing Company, Incorporated, 1st edition.

Jagadeesan, Nachikethas A., K. B. (2014). Software-defined networking paradigms in wireless networks: A survey. *ACM Comput. Surv.*, 47(2):27:1–27:11.

Kreutz, D., Ramos, F. M. V., Veríssimo, P., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *CoRR*, abs/1406.0440.

Kumar, S., Cifuentes, D., Gollakota, S., and Katabi, D. (2013). Bringing cross-layer mimo to today's wireless lans. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 387–398, New York, NY, USA. ACM.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA. ACM.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Moura, H., Bessa, G. V. C., Vieira, M. A. M., and Macedo, D. F. (2015). Ethanol: Software defined networking for 802.11 wireless networks. In *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, pages 388–396.

Nakagami, M. (1960). The m-distribution – A general formula of intensity distribution of rapid fading. In Hoffmann, W. C., editor, *Statistical Methods in Radio Wave Propagation*. Elmsford, NY.

Rappaport, T. (2001). *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

Sama, M. R., Contreras, L. M., Kaippallimalil, J., Akiyoshi, I., Qian, H., and Ni, H. (2015). Software-defined control of the virtualized mobile packet core. *IEEE Communications Magazine*, 53(2):107–115.

Santos, M. A. S., de Oliveira, B. T., Margi, C. B., Nunes, B. A. A., Turletti, T., and Obraczka, K. (2013). Software-defined networking based capacity sharing in hybrid networks. In *ICNP*, pages 1–6.

Schulz-Zander, J., Suresh, L., Sarrar, N., Feldmann, A., Hühn, T., and Merz, R. (2014). Programmatic Orchestration of WiFi Networks. In *USENIX ATC 14*, pages 347–358, Philadelphia, PA. USENIX Association.

Suresh, L., Schulz-Zander, J., Merz, R., Feldmann, A., and Vazao, T. (2012). Towards programmable enterprise wlans with odin. In *Proceedings of HotSDN '12*, pages 115–120, New York, NY, USA. ACM.

Yang, M., Li, Y., Jin, D., Zeng, L., Wu, X., and Vasilakos, A. V. (2014). Software-defined and virtualized future mobile and wireless networks: A survey. *CoRR*, abs/1409.0079.

Yap, K.-K., Sherwood, R., Kobayashi, M., Huang, T.-Y., Chan, M., Handigol, N., McKeown, N., and Parulkar, G. (2010). Blueprint for introducing innovation into wireless mobile networks. In *VISA '10*, pages 25–32, New York, NY, USA. ACM.

ONS: Simulador de Eventos Discretos para Redes Ópticas WDM / EON

Lucas R. Costa, Léia S. de Sousa, Felipe R. de Oliveira, Kaio A. da Silva, Paulo J. S. Júnior e André C. Drummond

[1]Departamento de Ciência da Computação – Universidade de Brasília (UnB), 70910-900, Brasília, Brasil

Abstract. *This paper presents the Optical Network Simulator - Wavelength Division Multiplexing / Elastic Optical Network (ONS-WDM/EON), a tool for optical networks evaluation. The ONS is a discrete event simulator which aims simulate traffic with arrivals and departures of connection requests from both WDM optical networks and the recent EON paradigm. ONS provides an environment for exploring the development of new Routing and Wavelength Assignment (RWA) protocols for WDM and new Routing, Modulation Level and Spectrum Allocation (RMLSA) algorithms for EON paradigm. We also present some examples of execution and results for simulator validation.*

Resumo. *Este artigo apresenta o Optical Network Simulator - Wavelength Division Multiplexing / Elastic Optical Network (ONS - WDM/EON), uma ferramenta para avaliação de redes ópticas. O ONS é um simulador de eventos discretos capaz de simular tráfego dinâmico com chegadas e partidas de requisições para ambos os paradigmas de redes ópticas WDM e o recente EON. O ONS proporciona um ambiente para a exploração do desenvolvimento de novos protocolos RWA (Routing and Wavelength Assignment) para WDM e algoritmos para RMLSA (Routing, Modulation Level and Spectrum Allocation) do paradigma EON. São apresentados exemplos de execução e resultados para validação do simulador.*

1. Introdução

A avaliação de desempenho dos sistemas de comunicação óptica é atualmente um desafio para os especialistas. Sua dinamicidade e complexidade não permite uma modelagem analítica precisa e a implementação de ambientes reais para medição muitas vezes é inviável por questões de custo e disponibilidade. A simulação é a alternativa disponível para atividades de teste, validação e avaliação de novos protocolos de controle para o ambiente de redes ópticas [Palmieri et al. 2009]. A simulação facilita a implementação de novos algoritmos fornecendo base para estudos e análises das características do plano de controle de maneira simples e escalável.

Várias ferramentas de simulação para redes ópticas de código livre podem ser encontradas na literatura. Dentre tais ferramentas, muitas derivam ou são módulos de simuladores de rede IP (*Internet Protocol*). Todavia, estas características trazem complexidades desnecessárias que podem ser abstraídas por simuladores exclusivos para rede óptica, permitindo uma maior velocidade de processamento por dispensar complexidades da camada IP. Por outro lado, também existem ferramentas proprietárias para tais serviços, entretanto suas características são limitadas e dificultam a manipulação por parte

do pesquisador [Palmieri et al. 2009]. Estas restrições reduzem ainda mais o número de simuladores desenvolvidos especificamente para redes ópticas de código aberto. O problema é agravado quando trata-se de uma ferramenta com foco no recente paradigma de redes ópticas elásticas (EON).

Este trabalho apresenta o ONS (*Optical Network Simulator - WDM / EON*)[1], um simulador de redes ópticas para os paradigmas WDM e EON. Inspirado no simulador WDMSim [Drummond], o ONS foi desenvolvido em Java e utiliza eventos discretos para simular requisições de tráfego de uma rede óptica WDM ou EON através de seus respectivos problemas de roteamento e alocação: RWA (*Routing and Wavelength Assignment*); e RMLSA (*Routing, Modulation Level, and Spectrum Allocation*)[2]. As funcionalidades projetadas permitem uma implementação fácil e ágil de novos algoritmos (RWA/RSA/RMLSA), fornecendo satisfatoriamente as principais métricas de desempenho para o cenário de tráfego de rede dinâmico, mesmo quando grandes topologias de rede são empregadas na simulação.

2. Trabalhos relacionados

A seguir, algumas ferramentas de simulação em redes ópticas de código aberto são destacadas na literatura. Entretanto, nenhuma delas é capaz de lidar simultaneamente com as tecnologias WDM e EON. O TONetS [Soares et al. 2007] (*Transparent Optical Network Simulator*) é um simulador para avaliação de redes ópticas WDM transparentes e proporciona a implementação de algoritmos RWA para soluções de técnicas de sobrevivência, estratégias de posicionamento de conversores de comprimentos de onda e políticas para reduzir injustiças no atendimento das conexões ópticas. O SimulNet [Palmieri et al. 2009], um ambiente de simulação de rede óptica especializada fornecendo a base para o estudo e análise das características de plano de controle com a implementação de algoritmos RWA, inclui suporte para redes translúcidas e possibilidade de extensão para o uso de algoritmos RWA que utilizam agregação (*grooming*). O SIMTON (*Simulator for Transparent Optical Networks*) [Chaves et al. 2010] é um simulador WDM projetado para redes ópticas transparentes. O SIMTON é mais completo que os demais mencionados, suas propriedades incluem características da camada física como atenuação, dispersão e ruído na fibra.

Com o surgimento das EONs e devido a impossibilidade de adaptação dos tradicionais simuladores WDM, novas ferramentas de simulação começaram a ser desenvolvidas para atender ao paradigma EON. Em [Asensio et al. 2013] os autores desenvolveram um simulador para EON baseado na biblioteca OMNeT++ [OMNeT++]. Esse simulador implementa um ambiente multicamada para reproduzir as arquiteturas da camada IP/MPLS (*Internet Protocol/Multiprotocol Label Switching*) e óptica, totalmente configurável e que permite implementar e testar novos algoritmos RSA em diversas arquiteturas de forma fácil e rápida. O CEONS (*Complex Elastic Optical Networks Simulator*) [Aibin and Blazejewski 2015] é um simulador de redes ópticas para resolver os problema RSA/RMLSA e de endereçamento de regeneradores. Segundo os autores, o simulador permite o desenvolvimento de algoritmos em diversas linguagens de programação, resolvendo os problemas em cenários de tráfego estático e dinâmico.

[1]O simulador ONS encontra-se disponível em http://comnet.unb.br/br/grupos/get/ons.

[2]Note que o problema RMLSA também inclui o problema RSA (*Routing and Spectrum Assignment*).

O FlexGrid [Moura and Drummond], um simulador de eventos discretos para EON, foi construído com base no simulador WDMSim [Drummond]. O ONS é inspirado em ambos, herdando a simplicidade de implementação para a criação de algoritmos RWA e RSA. A novidade está na implementação de algoritmos RMLSA e a execução de ambas as redes WDM e EON. Diversas métricas foram adicionadas e sua personalização ou expansão é facilmente aplicável. A Tabela 1 sumariza as propostas de código aberto apresentadas da literatura e suas características.

Tabela 1. Sumário dos simuladores apresentadas da literatura.

Simulador	Tráfego de Rede	Cenário de Rede WDM	Cenário de Rede EON	
		RWA	RSA	RMLSA
TONetS [Soares et al. 2007]	Dinâmico	Sim	Não	Não
SimulNet [Palmieri et al. 2009]	Dinâmico	Sim	Não	Não
SIMTON [Chaves et al. 2010]	Dinâmico	Sim	Não	Não
OMNeT++ [Asensio et al. 2013]	Dinâmico	Não	Sim	Não
CEONS [Aibin and Blazejewski 2015]	Estático Dinâmico	Não	Sim	Sim
WDMSim [Drummond]	Dinâmico	Sim	Não	Não
FlexGrid [Moura and Drummond]	Dinâmico	Não	Sim	Não
ONS (http://comnet.unb.br/br/grupos/get/ons)	Dinâmico	Sim	Sim	Sim

3. Simulador ONS

O simulador ONS pode ser logicamente separado em 4 módulos: (*i*) A configuração do ambiente de simulação; (*ii*) a geração dos eventos; (*iii*) a simulação em si; e (*iv*) a geração e avaliação dos resultados. A Figura 1 mostra o diagrama de funcionamento desses 4 módulos lógicos explicitados em detalhes a seguir:

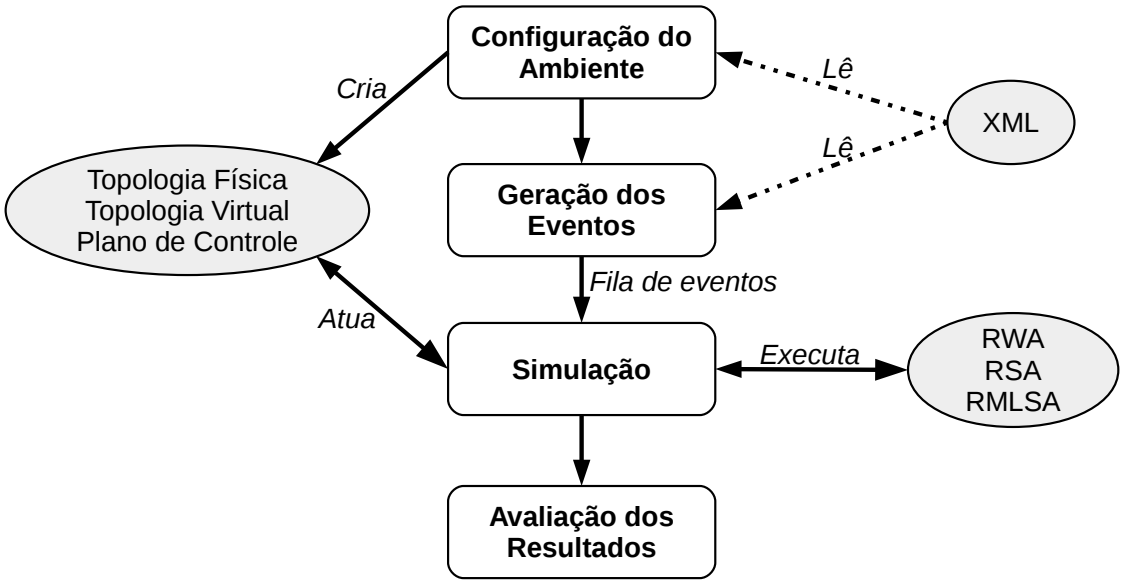


Figura 1. Diagrama de funcionamento do ONS.

Configuração do ambiente de simulação

A configuração do ambiente de simulação é inteiramente feita através da leitura de um arquivo XML[3] com os parâmetros de entrada. Cada *tag* fornece um tipo de configuração, entre elas estão: (*i*) O tipo de simulador (WDM/EON); (*ii*) o algoritmo (RWA/RSA/RMLSA); (*iii*) os dados relativos ao cenário de tráfego; e (*iv*) a configuração da topologia física (número de nós, enlaces, dispositivos e características da rede). Detalhes dessas informações podem ser consultadas no sítio do simulador.

Geração dos eventos da simulação

Cada evento é representado por um fluxo de dados com um nó de origem, um nó de destino e uma quantidade de banda passante. Conforme as informações existentes no XML (*tag* <*traffic*>), o simulador ONS gera os eventos de chegada e partida de requisição de conexão baseado em informações de carga, tempo médio de duração e taxa de transmissão. A taxa média de chegada é definida a partir da carga imposta, medida em Erlang. A carga pode ser definida como $E = T \times D$, onde E é a carga (Erlang), T é a taxa média de chegada e D é o tempo médio de duração de chamadas. O arquivo XML define as variáveis E e D, de forma que o tempo médio de chegada $\left(\frac{1}{T}\right)$ possa ser calculado. Em seguida, cada evento é gerado de acordo com um processo de Poisson, que define a distribuição dos tempos de chegada e partida, na qual o tempo médio é normalmente modelado por uma distribuição exponencial negativa baseado no tempo médio de chegada. Os nós de origem e destino dos eventos são escolhidos aleatoriamente com igual probabilidade (distribuição uniforme). As taxas de transmissão são definidas no XML e escolhidas aleatoriamente, sendo que cada taxa possui um tempo médio de duração definido e um peso para distribuição probabilística. A partir de então, uma fila de prioridades é utilizada para armazenar todos os eventos criados (chegadas e partidas), cuja ordenação é feita de acordo com o momento em que os eventos devem ocorrer durante a simulação.

Simulação

A simulação é realizada através da remoção dos eventos ordenados, a partir da fila gerada no módulo anterior. A cada evento na fila (chegada ou partida), o ONS chama o plano de controle que é responsável por registrar os eventos e executar um algoritmo RWA/RSA/RMLSA, baseado nas informações do estado da rede e do evento recebido, que por sua vez, toma a decisão de aceitar ou bloquear uma chamada. Note que, antes de aceitar uma chamada, o algoritmo deve resolver o problema de roteamento (RWA/RSA/RMLSA) através de interfaces do plano de controle, que controla a alocação de recursos na rede. O simulador deve validar os requisitos dos respectivos problemas e então é possível aceitar a chamada.

Avaliação dos resultados da simulação

Durante toda a simulação todas as ações são registradas em um arquivo de *trace*, tais como: (*i*) Chegadas e partidas de eventos; (*ii*) criação ou remoção de caminho óptico; e (*iii*) aceitação ou bloqueio de chamada. Além disso, o ONS ainda fornece uma interface para adição de código para cálculo de estatísticas, a qual pode ser personalizada de acordo com o foco da pesquisa. A interface é facilmente personalizável para a impressão dos resultados em tela (ex. formato CSV) ao termino de uma simulação, facilitando a geração automática de gráficos.

[3]O exemplo do arquivo XML encontra-se disponível em http://comnet.unb.br/br/grupos/get/ons.

3.1. Execução do simulador ONS

Por ser implementado em Java, a execução do ONS independe de plataforma, sendo necessário apenas que haja uma máquina virtual Java (JVM 7 ou superior) instalada no sistema operacional. O simulador deve ser executado a partir da linha de comando da seguinte forma:

```
you@computer:~$ java -jar ONS.jar
Usage:ONS simulation_file seed [-trace][-verbose][minload maxload step]
```

Os parâmetros obrigatórios do simulador empregados na linha de comando são os seguintes:

- **simulation_file:** o arquivo XML contendo os parâmetros da simulação;
- **seed:** trata-se de um número no intervalo [1-25] que define 25 diferentes conjuntos de sementes escolhidas internamente para maximizar a qualidade das sequências aleatórias utilizadas. Para a geração de resultados com intervalo de confiança em uma mesma simulação, se faz necessário a execução com diferentes conjuntos de sementes.

Os parâmetros opcionais do simulador são:

- **trace:** ativa a geração do arquivo *trace*;
- **verbose:** ativa o modo de geração de mensagens na tela;
- **minload maxload step:** permite a automação de diversas execuções de uma mesma simulação para uma faixa de cargas no intervalo [minload, maxload] e incremento [step].

A seguir são apresentados exemplos de uso do simulador ONS:

```
you@computer:~$ java -jar ONS.jar nsfnet.xml 1 > saida.txt
```

Neste exemplo, o simulador executará a simulação descrita no arquivo "nsfnet.xml" utilizando o primeiro grupo de sementes. A carga é única e está definida no aquivo XML.

```
you@computer:~$ java -jar ONS.jar nsfnet.xml 2 100 200 20 > saida.txt
```

Neste outro exemplo, o simulador executará a simulação descrita no arquivo"nsfnet.xml" utilizando o segundo grupo de sementes para uma faixa de cargas em Erlang que vai de 100 a 200, com incrementos de 20, ou seja, serão executados 6 simulações com as cargas [100, 120, 140, 160, 180, 200].

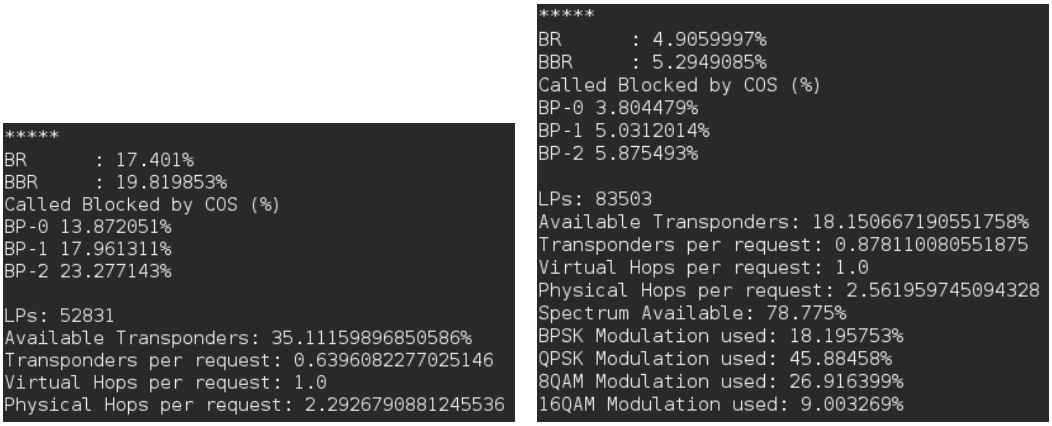
A Figura 2 apresenta exemplos de saída de uma simulação no cenário de rede WDM e EON. Note que esta saída pode ser personalizada. Observe que o simulador apresenta resultados de taxa de bloqueio (BR), taxa de bloqueio de banda (BBR), taxa de bloqueio por classe de serviço (COS), número de caminhos ópticos (LPs) estabelecidos na rede, percentual médio de transmissores disponíveis em toda a simulação, número de transmissores por requisição[4], e média de saltos virtuais e físicos por requisição. Para o cenário EON são adicionadas as métricas de percentual médio de espectro disponível em toda a simulação e percentual médio de uso por modulação.

4. Demonstração para o salão de ferramentas

Para demonstrar a utilização do simulador ONS é preterivelmente necessário uma máquina Dual Core com pelo menos 2GHz e 4GB de RAM, onde serão feitas as execuções de duas aplicações, sendo uma para WDM e outra para EON. Além disso, uma segunda máquina[5] será empregada para exibir detalhes da obtenção do simulador, instalação e

[4]Quando considerado técnicas de agregação de tráfego (*grooming*), um transmissor pode trafegar mais de uma requisição.

[5]Os autores poderão usar os notebooks próprios.



(a) (b)

Figura 2. (a) Exemplo de saída de uma simulação no cenário de rede WDM. (b) Exemplo de saída de uma simulação no cenário de rede EON.

funcionamento. Para demonstração da aplicação WDM será apresentado um algoritmo RWA explorando as possibilidades de determinação da rota, facilmente obtida a partir dos utilitários disponibilizados na ferramenta. Analogamente, a segunda aplicação a ser demonstrada é o algoritmo RMLSA para EON, que seleciona um nível de modulação dentre as oito disponíveis na ferramenta.

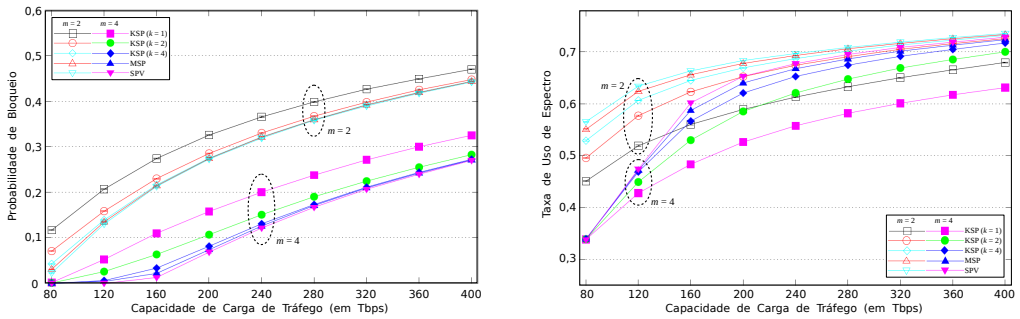
Quanto aos detalhes de instalação do simulador, orientações serão passadas sobre o seu funcionamento. Algumas configurações precisam ser ajustadas para cada tecnologia (WDM ou EON), como é o caso de componentes como os enlaces e nós ópticos. Outro ponto importante é o acompanhamento estatístico da simulação, que produz múltiplos dados a respeito do desempenho da rede, úteis para a caracterização dos experimentos.

5. Resultados e validação

Com o objetivo de validar a ferramenta de simulação foram realizados vários testes de verificação e análises de *traces* em diversas topologias de rede. Foram realizadas validações através de comparações com resultados de simulação de outros trabalhos, todas modeladas e testadas com sucesso.

Para a verificação do cenário de redes WDM, foi utilizado o simulador de redes ópticas WDMSim [Drummond]. Foram executados diversos algoritmos RWA em ambos os simuladores (ONS e WDMSim) e seus resultados obtiveram exatamente os mesmos valores para todas as métricas em comum.

Para a verificação do cenário em EON, foram realizadas comparações de resultados de outros trabalhos da literatura [Wan et al. 2012]. A Figura 3 mostra os resultados obtidos no ONS para a topologia NSFNET (14 nós e 44 enlaces) usada no mesmo artigo. Cada simulação foi realizada cinco vezes utilizando diferentes conjuntos de sementes (replicações independentes). Para os resultados apresentados foram calculados intervalos de confiança com 95% de confiabilidade. São demonstrados os valores de probabilidade de bloqueio e taxa média de uso do espectro para os dez algoritmos RSA apresentados em [Wan et al. 2012] nos níveis de modulação $m = 2$ (2 *bits* por símbolo) e $m = 4$ (4 *bits* por símbolo).



(a) (b)

Figura 3. (a) Resultados de simulação do ONS para o percentual de bloqueio. (b) Resultados de simulação do ONS para taxa de uso do espectro.

Os autores calculam a capacidade da carga de tráfego através da equação: $\rho \times \overline{C}$, onde ρ é a carga de tráfego em Erlang e $\overline{C}$ é a taxa média de bits igual a 105 Gbps. Pode-se observar na Figura 3(a) que à medida que a carga cresce, a probabilidade de bloqueio dos algoritmos aumenta para ambas as modulações. Observa-se ainda que os resultados apresentados aproximam-se bastante dos resultados disponibilizados de [Wan et al. 2012]. A Figura 3(b) mostra os resultados da taxa de uso do espectro. Observa-se que o uso do espectro cresce à medida que a carga de tráfego aumenta. Isso se deve ao fato que, o aumento da carga, implica na tentativa do estabelecimento de mais circuitos, que por sua vez, aumenta o nível de utilização do espectro na rede. Novamente os resultados obtidos no ONS são bastante próximos dos apresentados em [Wan et al. 2012]. A Tabela 2 mostra os resultados de alguns algoritmos e os seus respectivos tempos de execução em uma máquina Intel Core 2Quad 2.66GHz com 4GB de RAM. É possível observar que o ONS obteve um desempenho satisfatório mesmo com o aumento da carga de tráfego.

Tabela 2. Tempo médio de execução dos algoritmos no simulador ONS para 100.000 chamadas na topologia NSFNET (14 nós e 44 enlaces).

Algoritmo	Carga de Tráfego		
	80 Tbps	240 Tbps	400 Tbps
KSP(k=4)(m=2)	63,98s	71,62s	67,00s
KSP(k=4)(m=4)	53,70s	66,70s	67,57s
SPV(m=2)	741,42s	289,09s	207,96s
SPV(m=4)	725,12s	284,99s	208,14s

Diversos outros estudos e avaliações de métricas podem ser feitas com a utilização da ferramenta de simulação ONS. A título de exemplo, o ONS esta sendo utilizado pelo grupo de pesquisa em engenharia de tráfego da Universidade de Brasília (GET) para a resolução do problema RSA/RMLSA com o uso de técnicas de agregação óptica para a alocação dos recursos e previsões de novas funcionalidades, tais como roteamento multi caminho, técnicas para a proteção parcial e o roteamento e atribuição de espectro ciente da aplicação para Múltiplas Transferências de Dados em Massa (MBDT).

6. Considerações finais

Este artigo apresentou a ferramenta de simulação de redes ópticas ONS, desenvolvida com o objetivo de proporcionar um ambiente para a exploração de novos protocolos para os cenários de rede WDM e EON. Sua capacidade inclui desenvolver e avaliar novas pesquisas relacionadas aos problemas RWA, RSA e RMLSA no cenário de tráfego de rede dinâmico. O ONS proporciona uma implementação ágil e um desempenho satisfatório mesmo quando simulado em grandes topologias de rede, tornando-se uma ferramenta útil para a comunidade de pesquisa de redes ópticas.

Com o proposito de validar a ferramenta, foram realizados vários testes e validações parciais através de comparações com resultados de simulação de relevantes trabalhos da literatura. Atualmente, o simulador ONS está sendo utilizado e aprimorado pelo grupo de pesquisa em engenharia de tráfego da Universidade de Brasília (UnB). Sua infraestrutura já foi utilizada em diversos trabalhos e seminários da equipe.

Referências

[Aibin and Blazejewski 2015] Aibin, M. and Blazejewski, M. (2015). Complex elastic optical network simulator (ceons). In *17th International Conference on Transparent Optical Networks (ICTON), 2015*, pages 1–4.

[Asensio et al. 2013] Asensio, A., Castro, A., Velasco, L., and Comellas, J. (2013). An elastic networks omnet++-based simulator. In *15th International Conference on Transparent Optical Networks (ICTON), 2013*, pages 1–4.

[Chaves et al. 2010] Chaves, D. A. R., Pereira, H. A., Bastos-Filho, C. J. A., and Martins-Filho, J. F. (2010). Simton: A simulator for transparent optical networks. *Journal of Communication and Information Systems*, 25(1):1–10.

[Drummond] Drummond, A. C. WDMSim: WDM Optical Network Simulator. http://www.lrc.ic.unicamp.br/wdmsim/.

[Moura and Drummond] Moura, P. M. and Drummond, A. C. FlexGridSim: Flexible Grid Optical Network Simulator. http://www.lrc.ic.unicamp.br/FlexGridSim/.

[OMNeT++] OMNeT++. OMNeT++ Discrete Event Simulator. http://www.omnetpp.org/.

[Palmieri et al. 2009] Palmieri, F., Fiore, U., and Ricciardi, S. (2009). Simulnet: a wavelength-routed optical network simulation framework. In *IEEE Symposium on Computers and Communications, 2009. ISCC 2009.*, pages 281–286.

[Soares et al. 2007] Soares, A., aes, G. D., Giozza, W., and Cunha, P. (2007). Tonets: Simulador para avaliação de desempenho de redes Ópticas transparentes. *WPerformance, V Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, 27:581–594.

[Wan et al. 2012] Wan, X., Hua, N., and Zheng, X. (2012). Dynamic routing and spectrum assignment in spectrum-flexible transparent optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 4(8):603–613.

SCSimulator: An Open Source, Scalable Smart City Simulator*

Eduardo Felipe Zambom Santana[1,2], **Daniel Macêdo Bastista**[1], **Fabio Kon**[1], **Dejan S. Milojicic**[3]

[1]Department of Computer Science - University of São Paulo

[2]School of Engineering and Technology - Anhembi Morumbi University

[3]HP Laboratories - Palo Alto

{efzambom,batista,kon}@ime.usp.br,dejan.milojicic@hpe.com

Abstract. *Smart Cities, i.e., cities enhanced with a technological infrastructure that enables a more intelligent use and management of its resources, are currently seen as a powerful way of improving the quality of life of its citizens. Smart city platforms tailored at metropolises will be intrinsically very-large-scale systems; designing and developing such systems will be a daunting task. Also, deploying an infrastructure to test smart city systems is challenging due to costs, risks, and political difficulties. Therefore, being able to simulate the execution of smart city scenarios would be extremely beneficial for the advancement of the field. This paper presents a new open-source, large-scale smart city simulator capable of simulating different smart city scenarios and its performance and usability are also discussed.*

1. Introduction

Since 2009, most of the world population is living in cities [United Nations 2009] and current resources and infrastructure are hardly enough to cope with the increasing demand generated by population growth and geographic concentration [Caragliu et al. 2011]. Thus, making cities smarter can help optimize resource and infrastructure utilization in a more sustainable way. However, deploying the infrastructure to test Smart City systems will be a considerable problem due to costs, risks, and political difficulties.

There are already a few Smart City experimental testbeds such as SmartSantander [Jose et al. 2013], with 20 thousand sensors, and Padova Smart City [Zanella et al. 2014], with 3 thousand sensors. However, a Smart City has more components that are complicated to deploy in a testbed such as Smart Buildings, Smart Grids, and Vehicular Ah-Doc Networks (VANET). Another problem is that the current testbeds are deployed in small to medium cities. Deploying a sensor network in a huge city such as São Paulo, with 11 million inhabitants, is a daunting task.

An alternative to support large-scale tests and experiments is the use of simulators. The very large amount of actors in a Smart City environment (e.g., consider the 8 million cars and 11 million people in the city of São Paulo) requires a very-large-scale

*SCSimulator code, videos, and documentation are available in its page: https://github.com/ezambomsantana/smart-city-simulator

simulator. To be able to simulate such huge scenarios, we are developing SCSimulator to support tests of many Smart City complex scenarios such as traffic, energy, and disaster management. SCSimulator will certainly be very useful for many stakeholders such as city administrators, smart city software developers, and operators of city systems.

This paper is organized as follows. Section 2 presents related work. Section 3 describes the simulator functional and non-functional requirements. Section 4 presents the architecture and implementation of SCSimulator. Section 5 presents the SCSimulator performance and usability evaluation. Finally, Section 6 points out our conclusions and future work.

2. Related Work

In our literature and Web searches for Smart City simulators, we did not find any simulator that is capable of simulating large-scale and complex scenarios with multiple actors such as cars, buildings, people, and sensors. We found some specific simulators tailored at specific domains such as Smart Grids, VANETs, and car traffic, which we describe below.

DEUS (Discrete-Event Universal Simulator) is a discrete-event general purpose simulator, which was extended to simulate a Vehicular Ad-Hoc Network (VANET) [Picone et al. 2012]. In this Java-based, open-source simulator, it is possible to extend the base Node and Event model to implement specific actors to simulate entities such as cars, buildings, people, and sensors. Due to its architecture and non-parallel Java implementation, its scalability is weak, which was confirmed by experiments with almost 10 thousand nodes that we carried out.

Veins is a VANET simulator [Darus and Bakar 2013] integrated with OMNET++[1], a well-known discrete-event network simulator, and SUMO (Simulation of Urban Mobility)[2], a traffic simulator. In Veins, it is possible to simulate traffic scenarios such as accidents and traffic jams. In our experience with it, it was difficult for us to understand its code and architecture and running it in parallel mode was not trivial.

Siafu is a Java agent-based, open-source simulator [Europe 2007] used to simulate mobile events in a city. The simulator has a user interface to visualize simulation data and can export data sets. In Siafu, the agent creation is manual, so it is more appropriate for small, simple scenarios that can be visualized via a simple graphical interface.

3. Requirements

In this section, we present the functional and non-functional requirements that a Smart City simulator must handle.

3.1. Functional Requirements

To find the functional requirements for the initial version of our SCSimulator, we reviewed the literature on smart cities domains [Jose et al. 2013, Zanella et al. 2014]. We then selected the three most commonly explored domains in the scientific literature, which include the scenarios and actors described in the following.

[1]OMNET++ - https://omnetpp.org
[2]SUMO - http://sumo.dlr.de

- **Traffic:** It should be possible to simulate multiple traffic scenarios such as the impact of improving the quality of public transportation, the cause and effects of traffic jams, and the best routes for public transport systems. The main actors of these scenarios are vehicles, people, traffic lights, and flow sensors.
- **Resource Usage and Distribution:** In a Smart City, it is possible to manage the use of city resources such as water and electricity. It is possible to measure the amount of these resources at many levels such as buildings, streets, and neighborhoods. It should be possible to simulate the amount of water and electricity used by buildings as well as how these resources are distributed across the city. The main actors of these scenarios are Smart Buildings, Sensors, and Energy Stations.
- **Waste Management:** Many authors cite the management of waste services in the city as an important Smart City domain. It should be possible to simulate the use of a sensor network to notify when trash cans are full, visualization and control of trash vehicles, and prediction of when waste disposals will be full with the growth of population. The main actors of these scenarios are buildings (generate and receive trash), vehicles (trash truck), and sensors.

3.2. Non-Functional Requirements

The most important non-functional requirements necessary to implement a Smart City simulator are (1) enabling ease of use of the simulator and (2) the execution of very large simulations. Thus, the main non-functional requirements are:

- **Scalability:** To simulate Smart City scenarios, it will be necessary to manage millions of actors such as cars, people, buildings, and sensors. Therefore, the simulator scenarios have to scale from hundreds to millions of actors. To achieve this, distributed and/or parallel simulations will be mandatory.
- **Usability:** The simulator has to allow easy description of the simulated scenarios, enabling people that do not know the internal implementation of the simulator to develop scenarios with little effort. Thus, the programming model has to be intuitive and independent of the internal implementation of the simulator.
- **Extensibility:** It is unlikely that a simulator will provide all required features for Smart City simulations. The simulator has to be easily extensible, offering simple mechanisms for implementing new actors and changing their behavior, for implementing new metrics as well changing the behavior of the simulator itself. So, it is important not only that the simulator be open source but also that it is well documented and is implemented with good quality, extensible code.

4. SCSimulator

The goal of SCSimulator is to meet all the requirements stated in the previous section. To achieve that, our approach has been to reuse as much good quality code as we can. Thus, we based the project on an open-source, large-scale, discrete-event simulator called Sim-Diasca (Simulation of Discrete Systems of All Scales) [Song et al. 2011] developed in France by the EDF energy company.

Sim-Diasca is a general purpose simulator that has the goal of enabling very large-scale simulations. This simulator is implemented in Erlang, a functional language that facilitates the implementation of massively parallel and distributed applications. Moreover,

Sim-Diasca has a simple programming model enabling fast development of simulation scenarios. Our experiments with Sim-Diasca demonstrated that it scales much better and is much easier to use and extend that the other simulators mentioned in Section 2

Figure 1 presents the simulator architecture. The bottom layer is the Sim-Diasca simulator, responsible for the discrete-event activities such as Time Management, Random Number Generation, Deployment Management, and the Base Actor Models. The middle layer is the Smart-City Model we developed as part of our research, which implements the required actors for Smart City simulations such as vehicles, people, and sensors. The top layer comprises the scenarios that are implemented using the Smart City model.

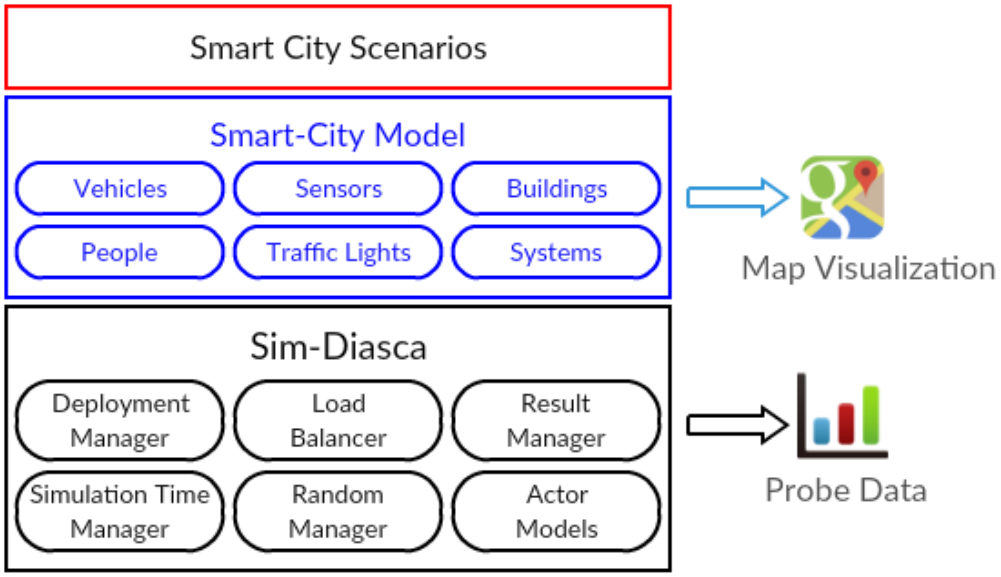


Figure 1. SCSimulator Architecture

With the SCSimulator, there are two ways to visualize the results of the simulation. First, it is possible to associate Probes with each actor and, anytime in the simulation, the actor sends data about its state to its probe. At the end of the simulation, a file with all the values passed to a probe is created. The simulator can also generate graphs with probe values over simulation virtual time. Figure 2 shows an example of a chart with the values of the probe of a sensor actor.

The second way to visualize the results of the simulation is via an animated GUI based on the GoogleMaps API. In this case, it is possible to visualize data gathered from the simulation while the simulation is running. Figure 3 shows an example of the simulation map, showing a simple simulation with a few dozen cars, buses, sensors, and traffic lights.

5. Simulator Evaluation

In this section, We compared SCSimulator with the three other simulators presented in Section 2. We compared three aspects of the simulators: Scalability, Programming Model, and Output.

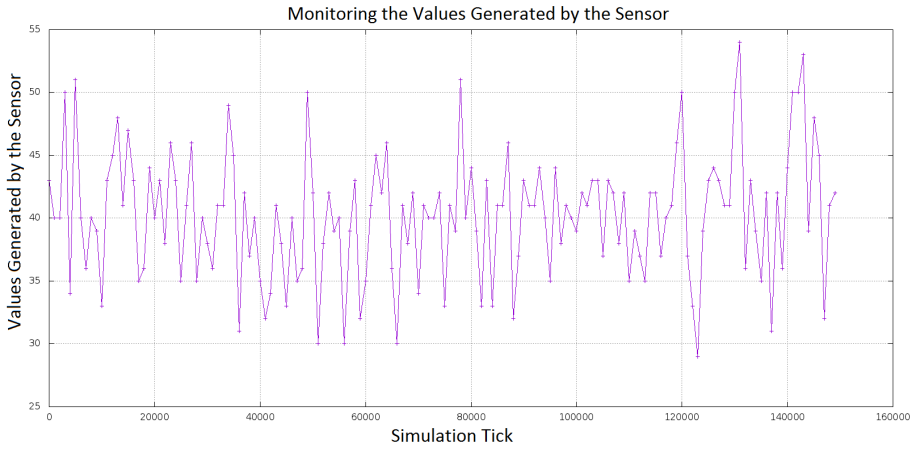


Figure 2. Probe generated data

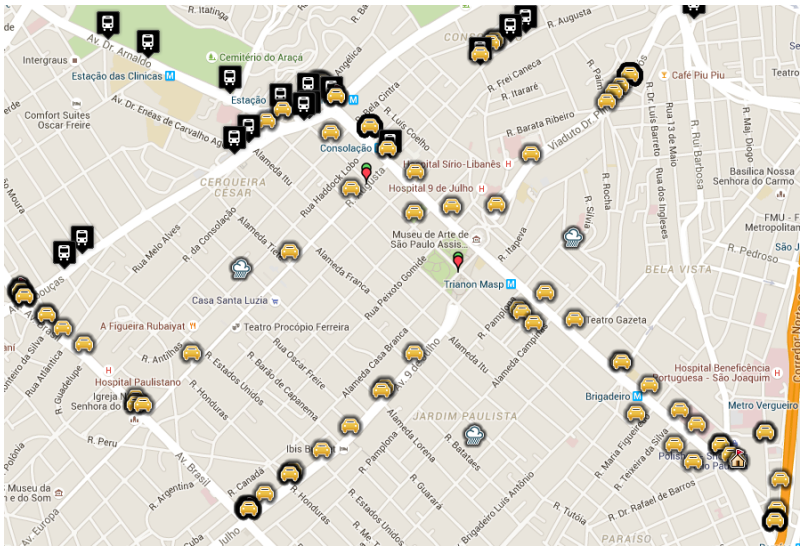


Figure 3. Animated Visualization with the SCSimulator

5.1. Scalability

To evaluate the scalability, we developed a scenario with some actors in SCSimulator. The initial configuration of the scenario had four buildings, four sensors, and three bus terminals. Cars came out of the buildings at a certain rate and buses left the terminals at another rate. We executed the simulation twice, the first with 500,000 simulated clock ticks and the second with 10 million simulated clock ticks. The first simulation took approximately 2,500 seconds and the second 100,000 seconds. To run the simulations, we used a machine with AMD FX6300 processor with 6 cores and 10 GB of memory running the GNU/Linux (Fedora 21) operating system.

Figure 4 shows the number of actors created during both simulations. The first simulation started with 6 actors and finished with 896. The second simulation also started with 6 actors and finished with 18128. The figure shows that the number of actors grew almost linearly with wall-clock time, indicating that the simulator could handle the load easily up to 18,000 actors and 10 million simulated clock ticks.

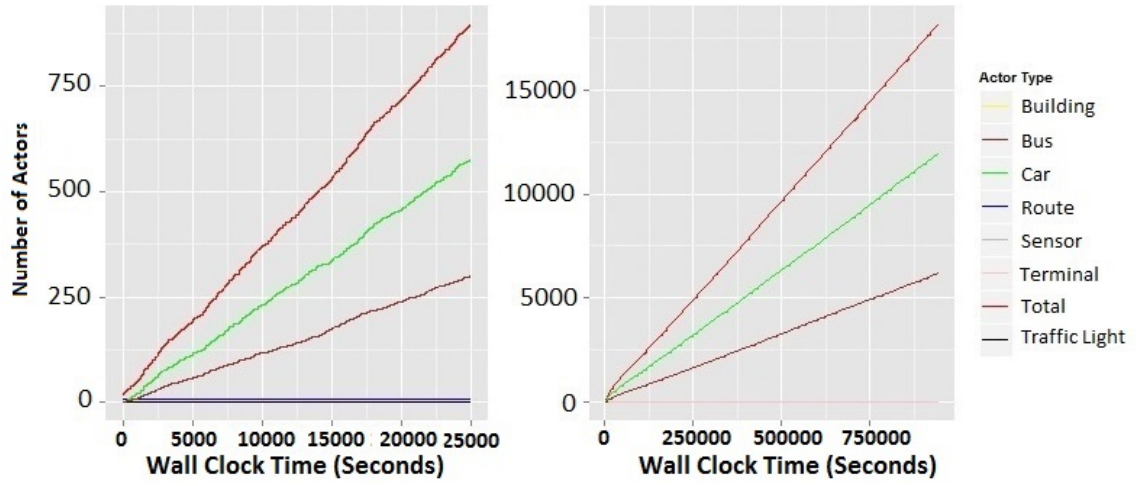


Figure 4. Number of actors over time

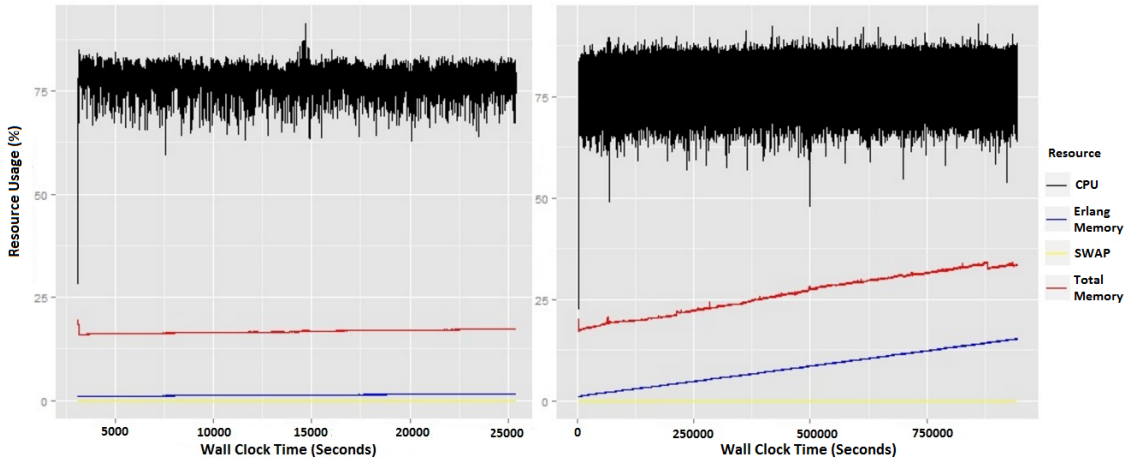


Figure 5. Computational resources used in the simulation over time

Figure 5 shows the resource consumption of both simulations. In the first simulation, the results show that resource usage almost did not change across the simulation. With 6 actors, the Erlang VM used 0.8% of the machine memory; with almost 900 actors, the amount of memory used increased to 2%. CPU usage also did not change very much over the simulation time; the chart shows that the average of the usage was almost the same during all the simulation. In the second simulation, the amount of memory occupied by the Erlang process also started in 0.6% and finished in 15.6%. The amount of memory used by the other processes of the machine increased almost at the same rate of the Erlang VM in both simulations. This occurred because creating an Erlang thread needs an OS action, and some processes create a file to write the values of the simulation.

These results show that the CPU is the bottleneck for the simulations up to the number of actors we experimented with, pointing to the necessity of parallelizing the execution of the simulation in more cores and/or machines.

We developed a similar simulation in DEUS and Siafu and executed on the same machine. DEUS stopped working with 7320 actors because of an out of memory error. In

Siafu we created a simulation with 384 actors but, as said in Section 2 it is hard to create a large scenario. These two simulators only execute sequentially. We did not present results obtained with Veins simulator because the simulator does not export the current number of actors in the simulation, but we could not create a very large simulation.

5.2. Programming Model

The programming model is important to evaluate the extensibility and usability of the simulator. The SCSimulator and DEUS have a very similar programming model. Both provide a base class (Actor in Sim-Diasca and Node in DEUS) that developers can extend to implement the simulation actors. The Siafu programming model is a little different, and the programmer has to understand all the code of the simulator. To create a traffic simulation in Veins no code is required, but if it is necessary to add new components to the simulator, it is necessary to change OMNET++ and SUMO and its communication. Therefore, SCSimulator and DEUS seem to be more easily extensible than Siafu and Veins.

To verify usability, it is important to analyze how to create the Smart City scenarios. In SCSimulator, an Erlang code describing the initial configuration of the simulation is required to define a scenario. Listing 1 presents an example of how to define the scenario actors. In the listing, a traffic light and a sensor actor are defined.

```
class_Actor:create_initial_actor( class_TrafficLight,
    [
        _TFNAME="traffic_light_1",
        _TFLAT=-23.562831,
        _TFLONG=-46.656866,
        _TFTIME=10
    ]),

class_Actor:create_initial_actor( class_Sensor,
    [
        _SName1="sensor_1",
        _SLat1=-23.570813,
        _SLong1=-46.656108,
        _SType1 = "temperature"
    ]),
```

Listing 1. SCSimulator Actors Definition

Veins and DEUS have a similar way of defining the scenario using XML files that describe the initial actors and their corresponding behavior what leads to an additional overhead for parsing. Siafu has a visual interface to define the scenarios, which is good and easy to define small simulations, but which makes impractical the creation of large simulations with a large number of actors.

5.3. Simulation Output

The SCSimulator has the two outputs already presented: the animated visualization based on Google Maps and the Probe datasets and graphs generated by the simulator. DEUS also uses Google Maps to present the results of the simulations but does not generate values to be analyzed after the simulation. Veins and Siafu have their own interfaces to show the simulation results; Siafu also generates datasets with values generated in the simulation for post-mortem analysis.

6. Conclusions and Future Work

This paper described the development of SCSimulator, a simulator that aims to advance the state of the art in the integrated simulation of Smart Cities, offering scalability and an easy programming model. In this first version of the simulator, we implemented actors of a Smart City environment such as Smart Buildings, Vehicles, and Sensors. The experiments showed that the simulator is scalable, a fundamental requirement to simulate Smart Cities. Compared to other simulators, SCSimulator is also easy to use and the results of the simulations can be obtained both by lists of values, graphs or an animated simulation with a GUI

In our ongoing work, we are experimenting with larger scenarios, going up to hundreds of thousands of actors; for that, we need to execute the simulator in larger machines, with more cores so as to better explore the parallelism supported by the Actor model of the Erlang language. In the long run, we intend to perform simulations with millions of actors but we anticipate that several challenges and bottlenecks will need to be resolved before we can achieve that. As future work, we intend to implement other Smart City scenarios such as disaster management and public transportation. Finally, we intend to perform a functional evaluation with city officials and public policy makers to validate the simulated scenarios and improve the simulator usability.

7. Acknowledgments

This work is funded by Hewlett-Packard Brazil.

References

[Caragliu et al. 2011] Caragliu, A., Del Bo, C., and Nijkamp, P. (2011). Smart cities in europe. *Journal of urban technology*, 18(2):65–82.

[Darus and Bakar 2013] Darus, M. Y. and Bakar, K. A. (2013). Congestion control algorithm in vanets. *World Applied Sciences Journal*, 21(7):1057–1061.

[Europe 2007] Europe, N. (2007). Siafu: An open source context simulator.

[Jose et al. 2013] Jose, A. G., Gutiérrez, V., Santana, J. R., Sánchez, L., Sotres, P., Casanueva, J., and Muñoz, L. (2013). Smartsantander: A joint service provision facility and experimentation-oriented testbed, within a smart city environment.

[Picone et al. 2012] Picone, M., Amoretti, M., and Zanichelli, F. (2012). Simulating smart cities with deus. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, pages 172–177. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Song et al. 2011] Song, T., Kaleshi, D., Zhou, R., Boudeville, O., Ma, J.-X., Pelletier, A., and Haddadi, I. (2011). Performance evaluation of integrated smart energy solutions through large-scale simulations. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 37–42.

[United Nations 2009] United Nations (2009). Urban and rural areas 2009.

[Zanella et al. 2014] Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of things for smart cities. *Internet of Things Journal, IEEE*, 1(1):22–32.

Salão de Ferramentas do SBRC 2016
Sessão Técnica 2
Redes Definidas por Software

OFSwitch13: Viabilizando o uso do OpenFlow 1.3 no ns-3

Luciano Jerez Chaves[1,2], **Islene Calciolari Garcia**[2], **Edmundo R. Mauro Madeira**[2]

[1] Depto. de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF)
Rua José Lourenço Kelmer, s/n. São Pedro, Juiz de Fora – MG.

[2] Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251. Cidade Universitária, Campinas – SP.

{lchaves, islene, edmundo}@ic.unicamp.br

Abstract. *The OpenFlow is a fast evolving technology that enables the implementation of Software Defined Networks. Despite the fact that the Network Simulator 3 (ns-3) already has a module for OpenFlow simulations, it is possible to note that the available implementation is very outdated. As many new major features were introduced up to the latest versions, it is desired to have them available for use. Thus, this paper presentes the OFSwitch13: a module to enhance the ns-3 with OpenFlow version 1.3 support. The paper describes the module design and implementation, and a case study scenario to demonstrate some of the available features.*

Resumo. *O OpenFlow é uma tecnologia em rápida evolução que permite a implementação de Redes Definidas por Software. Apesar do simulador de redes ns-3 já possuir um módulo para simulações com o OpenFlow, é possível observar que a versão existente está bastante ultrapassada. Como diversas novas funcionalidades foram introduzidas nas versões mais recentes, é desejável que elas estejam disponíveis para uso. Assim, este artigo apresenta o OFSwitch13: um módulo que habilita o ns-3 para simulações com o OpenFlow versão 1.3. O artigo descreve o projeto e a implementação do módulo, além de um estudo de caso para demonstrar algumas das funcionalidades presentes.*

1. Introdução

Como indicado pela [Open Networking Foundation 2012], os desafios na gerência de redes de estão se intensificando à medida que o número de dispositivos aumenta, sendo que atender às demandas do mercado é praticamente impossível com as arquiteturas tradicionais. Neste contexto, as Redes Definidas por *Software* (do Inglês *Software Defined Networking* – SDN) surgiram como um paradigma capaz de permitir a implantação de redes de maneira rápida e com baixo custo. Na arquitetura SDN, os planos de controle e de dados são desacoplados, a inteligência é logicamente centralizada em *software* enquanto a infraestrutura da rede é construída com comutadores (*switches*) programáveis de baixo custo. Com a inteligência centralizada é possível simplificar a tomada de decisões com base na visão global (ou parcial) da rede. O protocolo OpenFlow [McKeown et al. 2008] foi a primeira tecnologia desenvolvida especificamente para a realização do SDN, e tem sido adotado por fabricantes de equipamentos e também pela comunidade acadêmica.

Como se sabe, é custoso construir um ambiente de testes contendo vários equipamentos de rede para validar um determinado protocolo ou algoritmo. Nestes casos, ferramentas de *software* economizam muito dinheiro e tempo para realizar esta tarefa. Uma

escolha razoável seria utilizar um ambiente simulado para este fim, como o oferecido pelo simulador de redes *ns-3* [ns-3 2016]. O *ns-3* é um simulador de eventos discretos, desenvolvido para fins educacionais e de pesquisas, e distribuído como *software* livre. Simulações com o *ns-3* podem modelar comutadores OpenFlow através de um módulo disponível na distribuição padrão [OpenFlow for ns-3 2010]. Entretanto, este módulo implementa uma versão ultrapassada do protocolo [OpenFlow v0.8.9 2008]. Como diversas funcionalidades foram introduzidas nas versões mais recentes, é interessante que elas também estejam disponíveis para uso.

Neste cenário, este artigo apresenta o módulo `OFSwitch13` [OFSwitch13 2016] para o simulador *ns-3*. Este módulo permite a realização de simulações com o protocolo [OpenFlow v1.3.5 2015], equipando o simulador com o comutador de rede e com a interface para controlador. Com este módulo é possível interconectar nós da simulação através dos comutadores para enviar e receber tráfego de rede. Para gerenciar os comutadores, a interface para controlador pode ser estendida de maneira a implementar a lógica de controle desejada. O módulo depende de uma biblioteca externa chamada `ofsoftswitch13` [Fernandes e Rothenberg 2014], que de fato implementa e gerencia o caminho de dados do comutador OpenFlow. A biblioteca também possui a ferramenta `dpctl`, usada na criação de mensagens OpenFlow a partir de linhas de comando, e o código necessário para converter as mensagens OpenFlow para o formato de rede.

O restante deste artigo está organizado como segue: a Seção 2 descreve em linhas gerais o protocolo OpenFlow; a Seção 3 detalha a arquitetura do módulo `OFSwitch13` e seus componentes internos; a Seção 4 apresenta o cenário usado para demonstrar a ferramenta; e a Seção 5 conclui o artigo, apontando alguns trabalhos futuros.

2. O protocolo OpenFlow

O protocolo OpenFlow permite a construção de redes SDN através do controle refinado de tráfego em múltiplos equipamentos de redes, inclusive de diferentes fabricantes. O OpenFlow utiliza o conceito de fluxos para identificar um tráfego, com base em regras que podem ser configuradas pelo controlador dinamicamente. A especificação do OpenFlow cobre os componentes internos e as funções básicas do comutador, além do protocolo para comunicação entre comutador e controlador. A Figura 1 ilustra esta arquitetura.

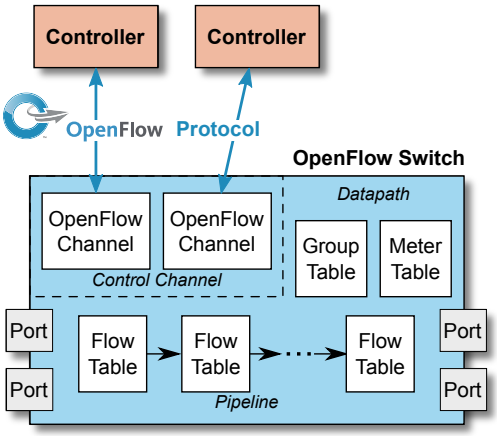


Figura 1. Arquitetura do comutador OpenFlow [OpenFlow v1.5.1 2015].

O caminho de dados do comutador OpenFlow consiste de um *pipeline* com uma ou mais *tabelas de fluxo*, que possuem entradas configuradas pelo controlador. Cada entrada de fluxo consiste de campos extensíveis para identificar o fluxo – *OpenFlow eXtensible Match* (OXM), contadores, e um conjunto de instruções para aplicar aos pacotes que coincidem com esta entrada. A busca começa na primeira tabela, em ordem de prioridade, e pode continuar pelas tabelas seguintes. Se um determinado pacote atende aos campos de uma entrada, então as instruções associadas à entrada são aplicadas ao pacote. As instruções podem modificar o processamento no *pipeline*, enviando o pacote para alguma das tabelas seguintes. Também podem conter ações que descrevem o encaminhamento do pacote, modificações em campos internos, e processamento por grupos e bandas de medição. A ação mais comum é a de saída, que encaminha o pacote para uma porta de saída. Outras ações disponíveis incluem a ação de grupo e a ação de medição, que direcionam o pacote para a *tabela de grupo* ou *tabela de medição*, respectivamente. Grupos representam conjunto de ações usados em semânticas de encaminhamento complexas, enquanto as entradas de medição permitem que o comutador controle a taxa de *bits* dos fluxos, limitando a banda passante em um valor pré-definido.

Para a comunicação entre comutador e controlador é utilizado o *canal OpenFlow*, protegido por *Transport Layer Security* (TLS), mas que também pode ser construído sobre TCP aberto. É através deste canal que o controlador configura as tabelas no comutador. O comutador deve suportar um único canal OpenFlow ou múltiplos canais, permitindo que diversos controladores compartilhem a tarefa de gerenciamento. O leitor pode consultar a especificação [OpenFlow v1.5.1 2015] para detalhes do protocolo.

3. O módulo OFSwitch13

A Figura 2 mostra a arquitetura do módulo `OFSwitch13`, destacando seus dois principais elementos: o *comutador* e a *interface para controlador* OpenFlow 1.3. Estes elementos interagem com a biblioteca externa `ofsoftswitch13`, e comunicam entre si pelo canal OpenFlow de controle, construído sobre protocolos e interfaces de rede disponíveis no *ns-3*. As subseções a seguir descrevem detalhadamente esta arquitetura.

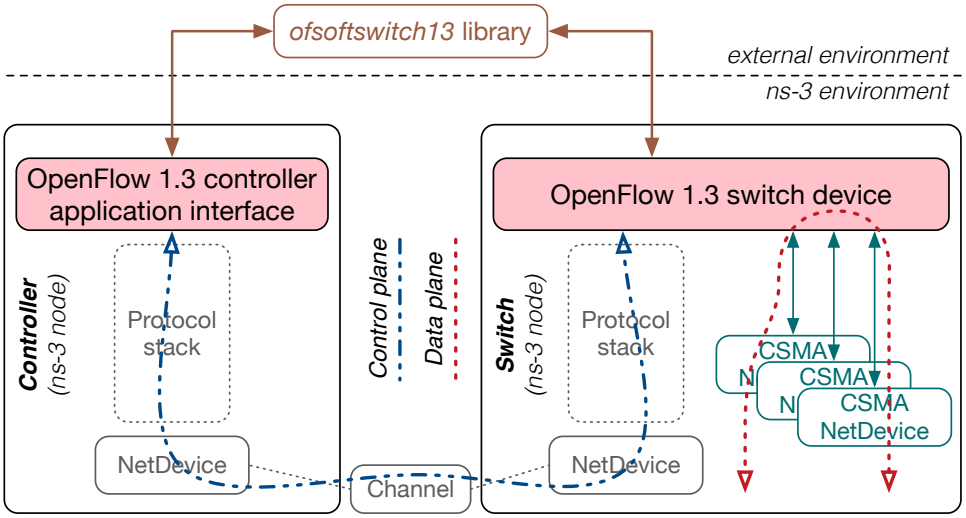


Figura 2. A arquitetura do módulo `OFSwitch13`.

3.1. Comutador OpenFlow 1.3

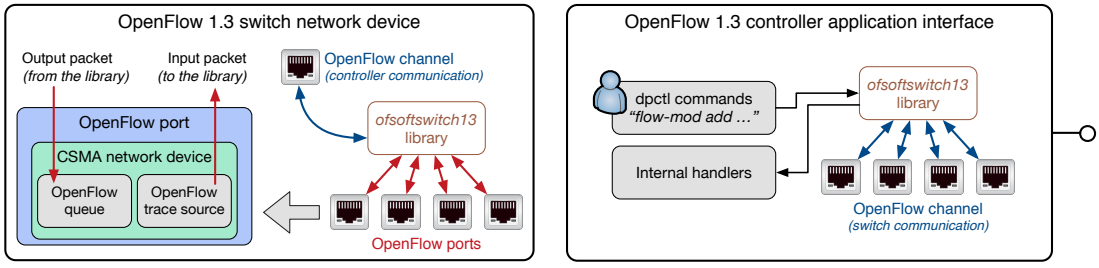
O comutador OpenFlow 1.3, ilustrado na Figura 3a, é usado para interconectar nós do *ns-3* através de interfaces do tipo *Carrier Sense Multiple Access* (CSMA), agindo como intermediário no encaminhamento dos pacotes. O comutador possui uma coleção de portas, cada uma associada à uma interface CSMA subjacente. Os pacotes entram nas portas através do *OpenFlow trace source* na interface CSMA. Esta nova fonte de rastreamento é configurada em modo promíscuo, e fornece os pacotes recebidos pela interface subjacente com seus cabeçalhos `Ethernet`. A inclusão desta fonte de rastreamento é a única modificação necessária ao código-fonte do *ns-3* para a integração com o `OFSwitch13`.

A implementação do caminho de dados do comutador (tabelas de fluxo, de grupo e de medição) são fornecidas pela biblioteca `ofsoftswitch13`. Por esta razão, os pacotes que entram no comutador são processados pelas funções da biblioteca. Para modelar o funcionamento do *hardware*, o módulo estima o tempo médio de busca nas tabelas de fluxo, tempo este usado para atrasar o início do processamento do pacote pela biblioteca. Para encontrar um atraso realista, o módulo considera que implementações reais do protocolo OpenFlow se baseiam em algoritmos de busca sofisticados. Como a maioria desses algoritmos realizam busca binária em árvores, a equação $k * log_2(n)$ é usada para estimar o atraso, onde k é uma constante que representa o tempo necessário para realizar uma única operação do tipo *Ternary Content-Addressable Memory* (TCAM) em *hardware*, e n é o número de entradas nas tabelas de fluxo do comutador.

Pacotes já processados pela biblioteca e marcados com uma ação de saída são enviados para a *fila OpenFlow* na porta de saída. O protocolo OpenFlow provê suporte limitado de Qualidade de Serviço (QoS) através de um mecanismo de filas com configurações distintas, onde uma ou mais filas podem estar associadas à uma porta, e as entradas de fluxos podem fazer o mapeamento dos pacotes nas filas. A fila OpenFlow possui uma coleção de filas internas, que são usadas como filas de prioridade para este fim.

3.2. Interface para controlador OpenFlow 1.3

A *interface para controlador* traz as funcionalidades básicas para a implementação de um controlador OpenFlow 1.3. Ela gerencia uma coleção de comutadores OpenFlow, conforme representado na Figura 3b. Para construir as mensagens de configuração no formato OpenFlow, a interface para controlador se baseia na ferramenta `dpctl`, disponível na biblioteca `ofsoftswitch13`. Com uma linha de comando de sintaxe simples, esta



(a) Comutador. (b) Interface para controlador.

Figura 3. Elementos da arquitetura do módulo `OFSwitch13`.

ferramenta pode ser utilizada para adicionar entradas nas tabelas, solicitar ao comutador por informações de estado, e mesmo modificar outras configurações do equipamento. Para as mensagens OpenFlow recebidas dos comutadores, a interface para controlador utiliza a coleção interna de manipuladores para tratar as mensagens de acordo com seu tipo. Alguns manipuladores podem ser sobrescritos pelos controladores que implementam a interface, de maneira a programar a lógica de controle desejada.

O módulo `OFSwitch13` também possui uma implementação desta interface com a lógica de um *controlador de auto-aprendizagem*. Este controlador funciona como um comutador tradicional, aprendendo em quais portas estão os nós da rede, e simplificando o processo de encaminhamento dos pacotes. Observe que o algoritmo de árvore geradora mínima não está implementado neste controlador, o que o impede de funcionar adequadamente em topologias de rede com ciclos.

3.3. Canal OpenFlow 1.3

O *canal OpenFlow* permite a comunicação entre o controlador e o(s) comutador(es). É através desta interface que o controlador configura e gerencia o caminho de dados. No módulo `OFSwitch13`, o canal OpenFlow é implementado por uma rede dedicada (fora de banda), usando as interfaces e canais disponíveis do *ns-3*. Pode-se optar por um canal OpenFlow compartilhado ou por conexões individuais entre cada comutador e o controlador. Este modelo provê conexões no plano de controle realista, que incluem atraso e, opcionalmente, modelos de erros. Ele também simplifica a análise do protocolo OpenFlow através do mecanismo de rastreamento do *ns-3*, que pode ser usado para gerar arquivos PCAP a serem lidos posteriormente por *softwares* convencionais.

Considerando que as mensagens no canal OpenFlow seguem o formato indicado na especificação, também é possível usar o módulo `TapBridge` do *ns-3* para integrar um *controlador externo*, rodando na máquina local, ao ambiente simulado. Entretanto, este caso de uso ainda não foi validado pelos autores.

3.4. Biblioteca ofsoftswitch13

O módulo `OFSwitch13` foi projetado para trabalhar em conjunto com a biblioteca `ofsoftswitch13`. É a biblioteca que, de fato, implementa o caminho de dados do comutador OpenFlow. Isto inclui as portas de entrada e saída, as tabelas de fluxo, de grupo e de medição. A ferramenta `dpctl`, disponível também na biblioteca, é usada para converter linhas de comando em mensagens no formato do protocolo OpenFlow 1.3. Para realizar a decodificação e a análise dos pacotes nas tabelas, a biblioteca utiliza o `NetBee` [NetBee 2016], que permite a descrição dos cabeçalhos de pacotes de redes através da linguagem *NetPDL*.

O código original do `ofsoftswitch13` foi ligeiramente modificado para a integração com o módulo `OFSwitch13` [OFSoftSwitch13 for ns-3 2016]. Entretanto, não houveram modificações na implementação do caminho de dados, que é mantida no repositório original e regularmente sincronizada com o repositório modificado. Para a integração, as portas de entrada e saída do `ofsoftswitch13` foram deixadas de lado, e a biblioteca foi modificada para enviar e receber pacotes diretamente para/do ambiente do *ns-3*. Para isso, todas as funções relacionadas com o envio e recebimento de pacotes foram anotadas como *símbolos fracos*, permitindo que o módulo sobrescreva essas funções

em tempo de ligação. Esta mesma estratégia foi usada para sobrescrever as funções relacionadas ao tempo, garantindo a consistência entre os tempos do simulador e da biblioteca. A integração também se baseia em chamadas de retorno usadas pela biblioteca para notificar eventos como descarte de pacotes por medições de banda, modificação no conteúdo do pacote por instruções nas tabelas de fluxo, pacotes duplicados por ações de grupo, e pacotes armazenados no comutador e enviados ao controlador.

Na tentativa de otimizar o processo de conversão de pacotes entre o formato otimizado do *ns-3* e a representação serial dos mesmos usada pela biblioteca, quando um pacote é enviado para processamento nas tabelas do OpenFlow, o módulo armazena temporariamente o pacote original do *ns-3*. Para os pacotes processados sem modificações no conteúdo, o comutador encaminha o pacote original do *ns-3* pela porta de saída específica. No caso de modificações no conteúdo, o comutador cria um novo pacote com o conteúdo modificado (copiando também os metadados do pacote). Esta abordagem se mostrou mas simples do que identificar quais foram as mudanças realizadas no pacote pela biblioteca.

3.5. Limitações atuais

Uma das limitações do módulo na implementação atual está na compatibilidade apenas com a plataforma GNU/Linux e o GNU *Compiler Collection* (GCC), por conta das modificações na biblioteca para integração com o simulador. Em relação ao protocolo OpenFlow 1.3, são duas as limitações principais: ausência de *conexões auxilares*, permitindo apenas uma única conexão entre o comutador e o controlador; e ausência de *suporte para múltiplos controladores*, forçando cada comutador a ser gerenciado por apenas um controlador. De acordo com a especificação do OpenFlow, conexões auxiliares poderiam ser criadas pelo comutador para aumentar o desempenho e explorar o paralelismo, enquanto múltiplos controladores aumentariam a confiabilidade já que o comutador poderia continuar operando mesmo que um controlador ou uma conexão apresentassem falhas.

4. Código, documentação e demonstração do módulo

Na página do projeto em `http://www.lrc.ic.unicamp.br/ofswitch13`, os visitantes irão encontrar uma breve descrição da ferramenta, além das informações necessárias para começar a usá-la. Existem ligações para os repositórios onde estão armazenados os códigos (Bitbucket para o código do módulo `OFSwitch13` e GitHub para a biblioteca `ofsoftswitch13` modificada), além da documentação que inclui o tutorial de instalação. Há também uma máquina virtual pré-configurada para facilitar o uso da ferramenta por novos usuários.

O módulo é licenciado sob a GNU GPL (mesma licença do simulador *ns-3*), enquanto a biblioteca modificada acompanha sua licença original BSD. A documentação do código-fonte gerada pelo Doxygen (classes e funções) pode ser consultada no endereço `http://www.lrc.ic.unicamp.br/ofswitch13/doc/html/`. O uso da ferramenta de *issues* no repositório do Bitbucket é encorajada para reportar erros e discutir sobre as funcionalidades do módulo.

Para a demonstração da ferramenta, a Figura 4 mostra a topologia de rede a ser utilizada. Ela representa a rede interna de uma organização, onde os servidores e os clientes estão separados por duas conexões de longa distância de 10 Mbps cada. Do lado servidor, o *comutador de borda* faz o papel do roteador ao responder as requisições dos

clientes e direcioná-las para o servidor interno adequado. Do lado cliente, o *comutador de agregação* é usado para gerenciar as conexões de longa-distância enquanto o *comutador cliente* concentra as conexões. O controlador de auto-aprendizado padrão é usado para gerenciar o comutador cliente, enquanto um novo *controlador de QoS* é responsável pelos demais comutadores. Os códigos do *ns-3* para esta demonstração estão disponíveis no diretório `examples/qos-controller/` do código-fonte do módulo `OFSwitch13`.

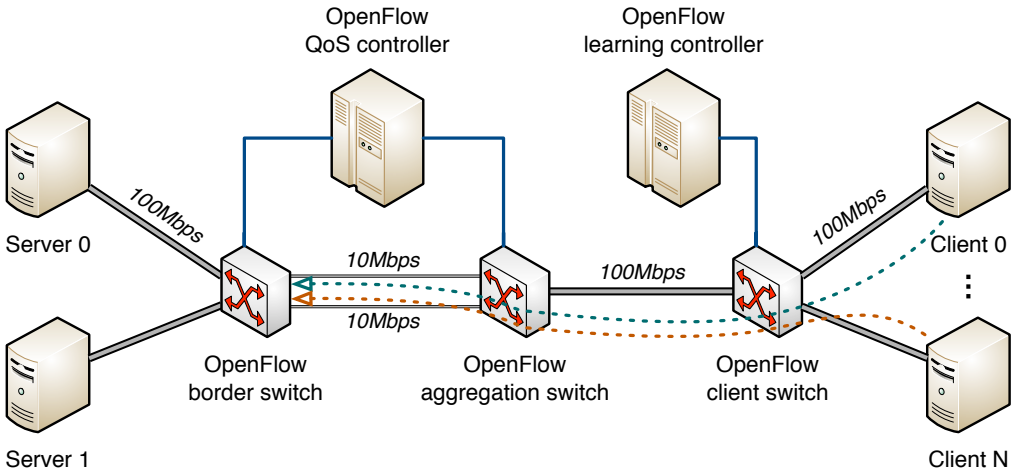


Figura 4. Topologia de rede para a demonstração.

A demonstração consiste em explorar algumas funcionalidades do OpenFlow 1.3 para implementar os seguintes mecanismos no novo controlador de QoS:

- **Agregação de enlaces:** Utilizar a tabela de grupos do OpenFlow (introduzida na versão 1.1), para realizar a agregação dos enlaces de longa distância. A agregação de enlaces pode ser usada para combinar diversas conexões de rede em paralelo, de maneira a aumentar a largura de banda. O objetivo é configurar os comutadores de agregação e de borda para dividir o tráfego pelos enlaces, aumentando a largura de banda entre clientes e servidores para 20 Mbps.
- **Balanceamento de carga**: Utilizar os campos extensíveis do OpenFlow (introduzidos na versão 1.2), para prover um serviço a partir de vários servidores, também conhecido como fazenda de servidores. O objetivo é configurar o comutador de borda para filtrar as requisições de ARP procurando pelo endereço IP do serviço e enviá-las ao controlador, para que este possa respondê-las de maneira a distribuir a carga de trabalho entre os servidores. Assim, o controlador instala as regras de fluxo nas tabelas do comutador de borda para que os próximos pacotes sejam diretamente encaminhados para o servidor escolhido, realizando também as modificações internas nos endereços de origem e destino dos pacotes de maneira que todo o processo seja transparente aos clientes.
- **Controle de fluxo**: Utilizar a tabela de medição do OpenFlow (introduzida na versão 1.3), para limitar a vazão das conexões de cada cliente para um valor pré-definido. O objetivo é configurar o comutador de agregação para controlar a vazão dos fluxos e evitar que eles sobrecarreguem as conexões de longa distância.

5. Conclusões e trabalhos futuros

Este artigo apresentou o módulo `OFSwitch13`, capaz de viabilizar simulações com o OpenFlow 1.3 no simulador de redes *ns-3*. O módulo é distribuído como *software* livre, e requer mudanças mínimas no código-fonte do *ns-3* para integração. Para demonstrar a ferramenta foram exploradas algumas características do OpenFlow 1.3, a saber: uso de tabelas de grupos para agregação de enlaces, o uso de campos extensíveis para balanceamento de carga, e uso de tabelas de medição para controle de vazão. O módulo `OFSwitch13` também foi usado para simulações no *ns-3* integrando o OpenFlow e redes *Long Term Evolution* (LTE), que são descritas em outro trabalho dos mesmos autores [Chaves et al. 2015]. Como trabalhos futuros, pretende-se eliminar as limitações atuais do módulo, além de implementar novas funcionalidades de QoS compatíveis com a recente camada de controle de tráfego do *ns-3*. Como este é um *software* livre, contribuições de desenvolvedores e usuários são sempre bem-vindas.

6. Agradecimentos

Os autores gostariam de agradecer ao Vítor M. Eichemberger e ao Eder L. Fernandes pelas contribuições, à Capes e ao CNPq pelo apoio financeiro (processo 118198/2014-9).

Referências

Chaves, L. J., Eichemberger, V. M., Garcia, I. C., e Madeira, E. R. M. (2015). Integrating OpenFlow to LTE: some issues toward Software-Defined Mobile Networks. Em *International Conference on New Technologies, Mobility and Security (NTMS)*, páginas 1–5. IEEE.

Fernandes, E. L. e Rothenberg, C. E. (2014). OpenFlow 1.3 software switch. Em *Simpósio Brasileiro de Redes de Computadores (SBRC)*, páginas 1021–1028. SBC.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., e Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

NetBee (2016). The NetBee library. Disponível em: `http://www.nbee.org/doku.php`.

ns-3 (2016). Network Simulator 3. Disponível em: `http://www.nsnam.org`.

OFSoftSwitch13 for ns-3 (2016). OpenFlow 1.3 software switch for ns-3. Disponível em: `https://github.com/ljerezchaves/ofsoftswitch13`.

OFSwitch13 (2016). The OpenFlow 1.3 module for ns-3. Disponível em: `http://www.lrc.ic.unicamp.br/ofswitch13`.

Open Networking Foundation (2012). Software-Defined Networking: The new norms for networks. ONF White Paper.

OpenFlow for ns-3 (2010). GSoC 2010 OpenFlow. Disponível em: `http://www.nsnam.org/wiki/GSOC2010OpenFlow`.

OpenFlow v0.8.9 (2008). OpenFlow switch spec. version 0.8.9. Open Networking Foundation.

OpenFlow v1.3.5 (2015). OpenFlow switch spec. version 1.3.5. Open Networking Foundation.

OpenFlow v1.5.1 (2015). OpenFlow switch spec. version 1.5.1. Open Networking Foundation.

OrchFlow - Uma Ferramenta para Orquestração de Múltiplos Controladores OpenFlow

Marcelo Frate[1], Marcelo K. M. Marczuk[2], Fábio L. Verdi[2]

[1]Instituto Federal de São Paulo (IFSP)
Boituva – SP – Brasil

[2]Universidade Federal de São Carlos (UFSCar)
Sorocaba, SP – Brasil

frate@ifsp.edu.br, marcelo.marczuk@dcomp.sor.ufscar.br, verdi@ufscar.br

Abstract. *The main purpose of the software defined networks is to centralize the control logic. However, it is possible to split this logic among two or more controllers in order to ensure scalability. The OpenFlow protocol defines the communication between switches and controllers, but does not provide for communication between controllers, required for any type of distribution in the control plane. It is necessary, therefore, to develop independent protocol solutions, capable of distributing this logic within the same administrative domain. In this scenario, we developed OrchFlow, a tool capable of orchestrating a software defined network with two or more OpenFlow controllers, enabling the management and monitoring of real-time topologies.*

Resumo. *O principal objetivo das redes definidas por software é a centralização da lógica de controle. Porém, é possível dividir esta lógica entre dois ou mais controladores com o intuito de garantir a escalabilidade. O protocolo OpenFlow define a comunicação entre switches e controladores, entretanto não prevê a comunicação entre controladores, necessária para qualquer tipo de distribuição no plano de controle. Faz-se necessário, portanto, o desenvolvimento de soluções independentes do protocolo, capazes de distribuir essa lógica dentro de um mesmo domínio administrativo. Neste cenário, desenvolvemos o OrchFlow, uma ferramenta capaz de orquestrar uma rede definida por software, com dois ou mais controladores OpenFlow, permitindo a gerência e o monitoramento da topologia em tempo real.*

1. Introdução e Motivação

As redes baseadas no protocolo IP [Postel et al. 1981] estão evoluindo, deixando a orientação ao hardware para a orientação ao software. Assim, *Software Defined Networking* (SDN) [Greenberg et al. 2005] e *Network Functions Virtualization* (NFV) [Cui et al. 2012] começam a se firmar como o futuro das infraestruturas de rede. Esses conceitos estão revolucionando a forma como se operam as redes pelo mundo, respondendo às mudanças da crescente demanda de tráfego pelos usuários e empresas. Para um *Data Center* o SDN responde principalmente à infraestrutura como uma opção para o aprovisionamento de serviços em nuvem, centralizando toda a administração dos equipamentos, definindo o funcionamento lógico da rede. Há, porém, razões para se usar

um plano de controle distribuído, o que dificulta a tarefa de gerenciamento dentro de um único domínio administrativo.

O objetivo deste trabalho é propor o uso de um software orquestrador para o aprovisionamento de serviços em redes definidas por software com múltiplos controladores OpenFlow, proporcionando ao administrador da rede uma visão global de seu domínio administrativo e um controle total das ações, independentemente de quantos e quais controladores serão utilizados. O OrchFlow permite, portanto, que os serviços fim a fim dentro do domínio administrativo continuem sendo fornecidos, assim como, serviços exclusivos podem ser criados para cada subdomínio.

2. Trabalhos Relacionados

Atualmente, diversas propostas para as redes SDN têm feito uso de múltiplos controladores com o objetivo de se obter um plano de controle descentralizado. O protocolo OpenFlow a partir de sua versão 1.2, torna possível a comunicação entre os *switches* e diversos controladores, possibilitando a criação de sistemas de *backups* e redundância. Entretanto, o OpenFlow não define a comunicação entre controladores, necessária para qualquer tipo de distribuição no plano de controle. Assim, é preciso que sejam desenvolvidas novas soluções independentes do protocolo capazes de distribuir essa lógica.

Alguns trabalhos encontrados na literatura têm por objetivo proporcionar a escalabilidade das redes. O **HyperFlow** [Tootoonchian and Ganjali 2010] é implementado como uma aplicação do controlador NOX [Gude et al. 2008], requerendo alterações de código no núcleo do controlador, para interceptar comandos e serializar eventos. **DISCO** [Phemius et al. 2014] procura compartilhar o estado da rede, mantendo a comunicação em cascata entre controladores vizinhos, apresentando também a necessidade de se reescrever o código do controlador Floodlight, tornando a proposta totalmente dependente deste. Por fim, **Onix** [Koponen et al. 2010] que executado em um *cluster* de um ou mais servidores físicos, utiliza um modelo de dados chamado *Network Information Base* (NIB), com uma contribuição relevante, mas, mesmo tendo por base a utilização de softwares de código aberto como o controlador NOX e do Apache ZooKeeper, foi desenvolvido em código fechado, o que impossibilita a integração e o desenvolvimento de novas aplicações.

Outras propostas buscam desenvolver os seus próprios controladores com protocolos próprios de comunicação para as interfaces leste-oeste, como o **Kandoo** [Yeganeh and Ganjali 2012], que utiliza os controladores de forma hierárquica passando a existir um controlador raiz no topo da rede interligando os diversos controladores locais. Embora esta abordagem tenha seus méritos, ela restringe ao controlador Kandoo, o que significa utilizar unicamente este controlador e atualmente o Kandoo suporta apenas a versão 1.0 do protocolo OpenFlow, dificultando o desenvolvimento de novos experimentos.

3. OrchFlow

As redes SDN concentram a inteligência através de um controlador, oferecendo um controle logicamente centralizado, possibilitando a existência de um ou mais controladores físicos. Entretanto, há diversas razões para se usar um plano de controle distribuído, tais como: administração, distribuição geográfica, escalabilidade, redução de latência entre

um *switch* e seu controlador, tolerância a falhas e balanceamento de carga dividindo a topologia da rede.

Porém, apesar de ser evidente a necessidade de subdividir a topologia para que cada parte seja gerenciada de maneira independente, há ainda a necessidade de gerenciamento e oferecimento de serviços globais que envolvem os diversos subdomínios. Um desses clássicos exemplos é o serviço de roteamento fim a fim.

Assim, desenvolvemos o OrchFlow, uma ferramenta que tem como objetivo funcionar como um *Middleware*, um software orquestrador para as redes SDN baseadas no protocolo OpenFlow [McKeown et al. 2008], capaz de receber solicitações de serviços através de uma interface Norte *(Northbound)*, processá-las e então mapeá-las através de uma interface Sul *(Southbound)*, de forma a prover o serviço solicitado em uma rede controlada por múltiplos controladores OpenFlow. Quando um determinado serviço é solicitado, o OrchFlow define como deverá ser o tratamento por parte dos diversos controladores até que sejam aplicadas as regras OpenFlow a todos os elementos de rede.

3.1. Funcionamento do OrchFlow

Como mostrado na Figura 1, a primeira camada da arquitetura contém os *switches*, formando toda a infraestrutura necessária para a comunicação entre os *hosts*. Para cada subconjunto de *switches*, definidos aqui como subdomínio, existe um controlador responsável pela configuração, por manter as informações da topologia e monitorar o estado da rede. Cada controlador está conectado ao OrchFlow através de uma interface Central.

O OrchFlow atua como um orquestrador, um agente integrador entre as diversas aplicações disponíveis na rede e os diferentes controladores OpenFlow, sob um mesmo controle administrativo, definido aqui como domínio administrativo. O OrchFlow possibilita a comunicação entre as diferentes aplicações através de uma interface Norte, capaz de receber solicitações e invocar serviços pré-determinados. Essas interfaces fazem parte de um único sistema WEB, utilizando *Representational State Transfer* (REST) [Fielding 2000], um protocolo que torna possível a troca de informações entre aplicativos e serviços WEB, pela qual são solicitados todos os recursos necessários para o estabelecimento de serviços fim a fim. Ao receber tais solicitações, o OrchFlow as processa e de forma orquestrada atua sobre cada um dos controladores conforme o subdomínio a ser alcançado, de duas maneiras possíveis:

Proativo: A aplicação inicia, via interface Norte, a solicitação de um serviço ao OrchFlow, que faz toda a orquestração necessária aos controladores via interface Central. Neste modo, toda a programação é executada pelo OrchFlow e transferida aos controladores via interface REST, que aplica todas as regras de fluxos necessárias para o estabelecimento de serviços fim a fim, diretamente aos *switches* envolvidos. Um serviço ou ação do administrador da rede solicita ao OrchFlow que encaminhe os parâmetros aos controladores para que sejam gravados diretamente na tabela de fluxos dos *switches*, possibilitando assim, que todos os fluxos sejam encaminhados diretamente às portas de destino. Os parâmetros a serem enviados aos *switches* são aqueles padronizados pela especificação OpenFlow, que permitem um controle total dos fluxos conforme as políticas do administrador do domínio.

Reativo: No modo reativo, as regras são criadas e enviadas para os controladores, porém, diferentemente do modelo Proativo, elas não são implantadas imediatamente nos

switches. Tais regras serão implantadas em resposta a um evento *packet-in*. Este evento é enviado através da interface Sul (OpenFlow) para o controlador do subdomínio. O controlador fará então uma busca em sua programação e, caso haja uma correspondência, ele adiciona as regras nas tabelas de fluxos de todos os *switches* de seu subdomínio, necessários para o estabelecimento do serviço solicitado.

Para que o modelo Reativo funcione, o OrchFlow disponibiliza uma interface que possibilita ao administrador da rede criar a programação de forma centralizada, descarregando as regras nos controladores conforme o serviço a ser fornecido.

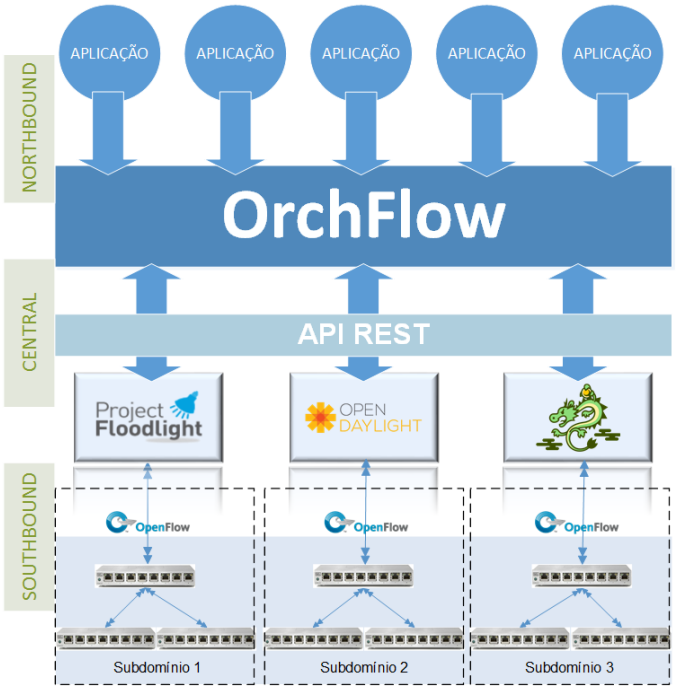


Figura 1. Arquitetura do OrchFlow.

Para efeito de testes e funcionamento exclusivo com o OrchFlow, foram desabilitados todos os módulos *forward* dos controladores de forma que não interfiram no funcionamento da rede. Assim, para todo e qualquer fluxo que chegue à rede, é necessário que haja uma regra correspondente na tabela de fluxos (modo Proativo). Caso não haja, os controladores devem possuir uma programação prévia para determinarem a configuração a todos os *switches* envolvidos no serviço solicitado (modo Reativo). Porém, caso um fluxo novo chegue à rede e o mesmo não encontrar correspondência na tabela de fluxos ou não houver uma programação que possa tratá-lo de forma correta, este será descartado.

4. Implementação e Validação

A topologia proposta tem, para efeito de testes, um domínio administrativo composto por três subdomínios, com um controlador OpenFlow em cada subdomínio. Cada controlador é responsável pelo controle e gerenciamento de um único subdomínio e cabe ao OrchFlow a orquestração, gerencia e a visão completa do domínio administrativo.

A Figura 2 ilustra a topologia utilizada nos testes do OrchFlow. Quatro máquinas virtuais (VM) são utilizadas:

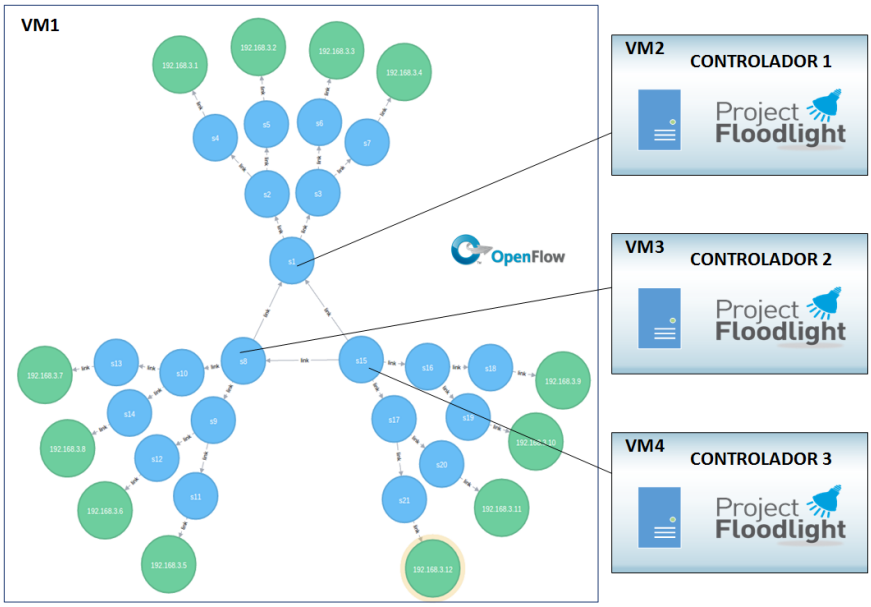


Figura 2. Topologia de rede.

VM1: Uma rede é emulada através do emulador Mininet [Lantz et al. 2010], contendo três redes interligadas em forma de anel, com 7 *switches* cada em forma de árvore e 4 *hosts*;

VM2, VM3 e VM4: Para o atual estágio de desenvolvimento e testes, temos em cada VM um controlador Floodlight funcionando e pronto para receber os eventos vindos dos *switches* das redes emuladas na VM1;

O OrchFlow foi desenvolvido e implementado em linguagem Java e utiliza um banco de dados não relacional, o Neo4j [Eifrem 2009]. As três redes estão definidas e interligadas através de *links* específicos, denominados aqui de *links* externos, utilizando ainda *switches* virtuais *OVS-Switch* e o protocolo OpenFlow na versão 1.3. Para a realização dos testes, escolhemos um dos serviços de rede mais conhecidos, o serviço de roteamento fim a fim. Neste artigo, cada subdomínio utiliza o controlador Floodlight. O uso de diferentes controladores será explorado em trabalhos futuros.

O desenvolvimento do OrchFlow busca proporcionar a integração, a gerência, a administração e a programação para as redes SDN baseada em múltiplos controladores OpenFLow. Para isso ele deve ser capaz de ler e gravar as diversas programações em cada um dos controladores, de forma a garantir o funcionamento dos serviços entre quaisquer *hosts* pertencentes ao mesmo domínio administrativo, mesmo estando localizados em diferentes subdomínios.

4.1. Banco de Dados - Neo4j

Para que haja a correta leitura e consolidação dos dados é necessário armazená-los de forma eficiente e bem estruturada. Assim, foi escolhido o Neo4j [Eifrem 2009], um banco de dados orientado a grafos baseado em Java que oferece persistência, alto desempenho, escalabilidade e possui uma comunidade ativa e uma boa documentação. Neste trabalho, o Neo4j recebe os dados de cada um dos *hosts* e *switches* e representa-os como nós, obtém os dados dos *links* e cria um relacionamento entre eles, possibilitando ao administrador

obter uma visão completa de todo o domínio administrativo, através de uma interface gráfica.

Para que o sistema funcione, é necessário o reconhecimento de toda a topologia da rede. O OrchFlow obtém estes dados de cada controlador através da API REST já definida em cada controlador. Note que a maioria, se não todos os controladores, possuem uma chamada em REST para obtenção da topologia. Sendo assim, o OrchFlow se aproveita desta interface já existente para construir a topologia completa do domínio administrativo e armazená-la no Neo4J.

Uma vez que se tenha armazenado todos os dados relacionados à rede, pode-se obter o melhor caminho entre dois *hosts* e implementar o serviço de roteamento fim a fim. Neste trabalho, todas as rotas definidas pelo administrador ou por solicitação de alguma aplicação ao OrchFlow são criadas tomando por base o algoritmo de Dijkstra [Dijkstra 1959], que faz uma busca pelo caminho mais curto e determina aos controladores que criem os fluxos conforme o resultado encontrado. Assim, qualquer rota, mesmo as que passam por dois ou mais subdomínios, serão definidas por seus controladores conforme o que foi definido pelo OrchFlow.

4.2. Módulos ARPReply e Reactive

O modelo tradicional de abordagem para tratamento do pacote *ARP REQUEST* visa o envio da mensagem em *broadcast*. Nesse modelo, quando um *switch* não conhece o destino, realiza *broadcast* para todas as portas de saída, menos aquela onde recebeu a mensagem. Neste trabalho, desenvolvemos um módulo responsável por responder às mensagens de ARP dentro do domínio administrativo. Este módulo visa a criação de um pacote ARP REPLY para que a requisição ARP REQUEST seja respondida pelo controlador. Nela, o controlador usa todas as estruturas que possui para montar a resposta de um pacote ARP e enviá-lo de maneira transparente ao *host* solicitante.

Nesse sentido, cada controlador reconhece a topologia de seu subdomínio e a repassa ao OrchFlow. De posse das informações de todo o domínio administrativo cria-se então uma tabela de conversão MAC/IP. Tal tabela é então enviada para cada controlador através de uma interface REST estendida. O módulo ARPReply previamente instalado irá armazenar esta tabela para montar respostas às requisições ARP. Assim, mesmo que um *host* solicite um serviço fora de seu subdomínio, o controlador já terá os dados necessários para devolução da correta mensagem de retorno do protocolo ARP. Essa solução traz um ganho à rede, eliminando o *broadcast* de mensagens ARP.

Além do ARPReply, também desenvolvemos o módulo *Reactive* responsável por receber as regras do OrchFlow e armazená-las em uma estrutura interna. Estas regras serão posteriormente utilizadas para tratamento de eventos de *packet-in* originados na rede. Este modo de ação permite que se use as tabelas de fluxos de maneira mais eficaz diminuindo as entradas na memória *Ternary Content Access Memory* (TCAM) de cada *switch* na rede.

4.3. Interface

A ferramenta apresenta uma interface simples onde os serviços a serem aprovisionados no domínio administrativo podem ser solicitados. Inicialmente, os controladores preci-

sam ser cadastrados. Após isso, regras OpenFlow podem ser definidas e aplicadas nos subdomínios através de seus respectivos controladores.

Uma vez cadastrados todos os controladores, o sistema estará pronto para uso, carregando a página conforme a Figura 3. Nesta página é possível analisar a topologia da rede, alterando cores, mudando posições de nós e *links* dentre outras opções.

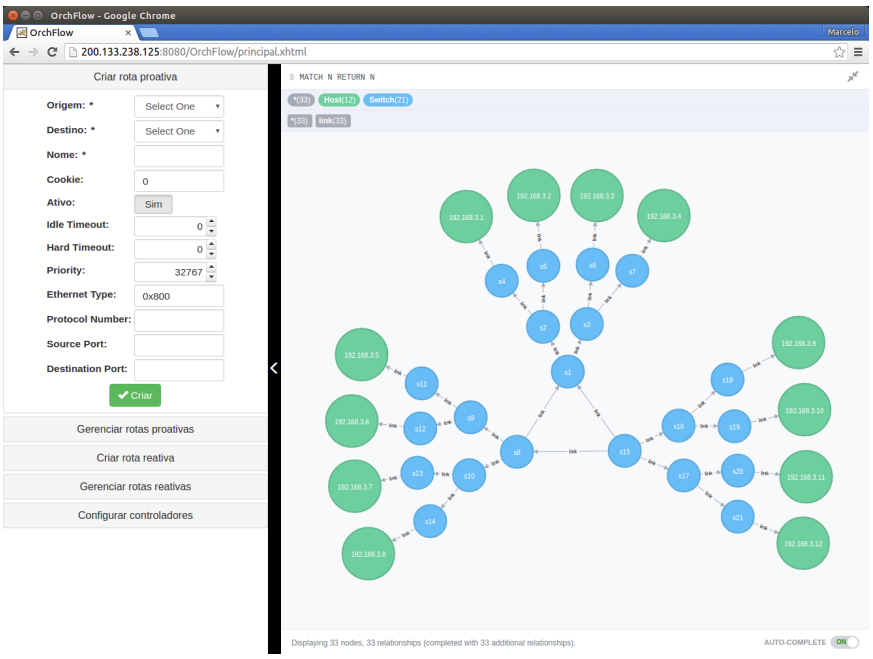


Figura 3. OrchFlow: Programação e Uso.

Do lado esquerdo da interface é possível identificar os dois modos de operação do OrchFlow no menu de opções: modos proativo e reativo. Para ambos os modos o procedimento para solicitar o serviço de roteamento fim a fim é o mesmo. O que diferencia um modo do outro é a forma pela qual serão criadas as regras. No modo proativo as regras são enviadas aos controladores que, por sua vez enviam imediatamente para todos os *switches* envolvidos no serviço solicitado. No modo reativo as regras são enviadas para cada controlador, porém ficam armazenadas em uma estrutura interna e serão enviadas aos *switches* apenas em eventos de *packet-in* ocorridos nos controladores.

A descrição da demonstração planejada para este trabalho encontra-se no manual do usuário que assim como o código fonte dos módulos necessários para Floodlight estão disponíveis em:

`https://github.com/marcelofrate/OrchFlow/`

5. Conclusão

O OrchFlow possibilita que as aplicações tirem proveito de todas as funcionalidades dos recursos físicos da infraestrutura de uma rede baseada em software de forma automatizada. Neste artigo, esperamos demonstrar que a integração entre controladores diferentes e a sua orquestração responda aos problemas de escala que atualmente existem nas redes SDN baseadas no protocolo OpenFlow.

O OrchFlow é uma ferramenta capaz de automatizar o processo de programação de múltiplos subdomínios, integrados a um único domínio administrativo, com uma visão global, centralizada, possibilitando o funcionamento nos modos proativo e reativo.

Os trabalhos futuros incluem o uso de diferentes controladores já que neste artigo apenas o controlador Floodlight foi usado. Além disso, testes de escalabilidade deverão ser realizados a fim de avaliar o potencial do OrchFlow em gerenciar domínios administrativos grandes, com centenas de elementos de rede e *hosts*. Possiveis questões de segurança, redundancia ou falta de comunicação entre os controladores e o OrchFlow poderão ser tratados também.

Referências

Cui, C., Deng, H., Telekom, D., and Michel, U. (2012). Network Functions Virtualisation. *Citeseer*, (1):1–16.

Dijkstra, E. (1959). A note on two problems in connexion with graphs:(numerische mathematik, ¯1 (1959), p 269-271).

Eifrem, E. (2009). Neo4j-the benefits of graph databases. *no: sql (east)*.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. *Building*, 54:162.

Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., and Zhang, H. (2005). A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54.

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). Nox: Towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., Others, and Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. *OSDI, Oct*, pages 1—-6.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. . . . *Workshop on Hot Topics in Networks*, pages 1–6.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Phemius, K., Bouet, M., and Leguay, J. (2014). DISCO: Distributed multi-domain SDN controllers. *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*.

Postel, J. et al. (1981). Rfc 791: Internet protocol.

Tootoonchian, A. and Ganjali, Y. (2010). Hyperflow: a distributed control plane for openflow. *Proceedings of the 2010 internet network . . .* , pages 3–3.

Yeganeh, S. H. and Ganjali, Y. (2012). Kandoo: a framework for efficient and scalable offloading of control applications. *Proceeding HotSDN '12 Proceedings of the first workshop on Hot topics in software defined networks*, pages 19–24.

TinySDN: Enabling TinyOS to Software-Defined Wireless Sensor Networks

Bruno T. de Oliveira[1], **Cíntia B. Margi**[1]

[1]Escola Politécnica – Universidade de São Paulo
Departamento de Engenharia de Computação e Sistemas Digitais

{brunotrevizan,cintia}@usp.br

Abstract. *Software-Defined Networking (SDN) has been envisioned as a way to reduce the complexity of network configuration and management, enabling innovation in production networks. While the first SDN approaches focused on wired networks, there were proposals specifically for Wireless Sensor Networks, but none of them targeted the TinyOS environment. This paper presents the TinySDN tool, a TinyOS-based Software-Defined Networking implementation. It comprises two main components: the SDN-enabled sensor node, which has an SDN switch and an SDN end-user device, and the SDN controller node, where the control plane is programmed. The SDN features have been implemented and tested in a real mote, and will be shown in the demo proposed.*

1. Introduction

Wireless Sensor Networks (WSN) have been used to support several different applications, mainly related to monitoring and detection. Nodes in a WSN are typically battery-powered and resource constrained (i.e. limited amount of memory, processing and communication), and communicate through a multihop ad hoc network [Culler et al. 2004].

Software-Defined Networking (SDN) has been envisioned as a way to reduce the complexity of network configuration and management, initially focused on wired networks. The main approach to SDN is OpenFlow[1], which focus on wired networks.

SDN for WSN imposes different challenges and requirements, but provides several opportunities. Among the challenges we highlight limited resources of WSN nodes: energy, processing, memory, and communication. Requirements are related to the applications characteristics (e.g. data frequency and size), as well to the nodes behavior due to duty-cycles, operating systems and programming approach. On the other hand, opportunities provided by SDN include: to improve resource reuse, to implement node retasking, node and network management, as well as to enable experiments with new protocols, and to ease transition to standard protocols for deployed networks in WSN and Internet of Things context [de Oliveira et al. 2015].

The main proposals in the literature concerning SDN for WSN are: Flow-Sensor [Mahmud and Rahmani 2011], Sensor OpenFlow [Luo et al. 2012] and SDWN [Costanzo et al. 2012]. Flow-Sensor [Mahmud and Rahmani 2011] proposes sensor nodes with the main features of OpenFlow. Sensor OpenFlow and SDWN propose a clear separation between data plane and control plane, a centrally controlled communication protocol between these two planes and some data plane features.

[1]https://www.opennetworking.org/sdn-resources/openflow/

Gante et al. [Gante et al. 2014] propose a framework to apply SDN to WSN management. Besides the benefits highlighted by previous works, authors include accurate localization and topology discovery as advantages of SDN usage in WSN. The SDN controller should be implemented as part of the WSN sink.

TinySDN [de Oliveira et al. 2014] is a flow-ID-based approach that improves on previous work by addressing the use of multiple SDN controllers and by discussing common WSN characteristics, such as (i) in-band control, opposed to SDN implementation in wired networks that can leverage from out-band control; (ii) higher communication latency; (iii) smaller link layer frames; and (iv) limited energy supply. Furthermore, the paper describes the protocol for communication between controllers and WSN nodes, and discuss implementation issues and approaches for TinyOS [Hill et al. 2000].

SDN-WISE [Galluccio et al. 2015] defines simple mechanisms for the definition and handling of the Flow Table that make it stateful, pursuing two goals: (i) to reduce the amount of information exchanged between sensor nodes and the SDN network controller, and (ii) to make sensor nodes programmable as finite state machines, so enabling them to run operations that cannot be supported by stateless solutions.

Given the lack of available SDN implementations for TinyOS, a widely used WSN operating system, we present TinySDN tool. It enables the SDN paradigm, making testing new protocols easier and providing multiple controllers for software-defined WSN. Each sensor node is composed of an SDN switch and an end-user device, which we call *SDN-enabled sensor node*. The control plane is programmed through a central component, which we call *SDN controller node*, thus centralizing the control plane of the WSN.

2. The TinySDN Tool

This section presents the TinySDN architecture overview by describing its components and protocols.

Figure 1 depicts the two TinySDN types of nodes: *SDN-enabled sensor node* and *SDN controller node*. Each *SDN-enabled sensor node*, where data plane components are installed, connects through multi-hop wireless communication to an *SDN controller node*, where the control plane logic is executed, allowing the interaction between the two planes.

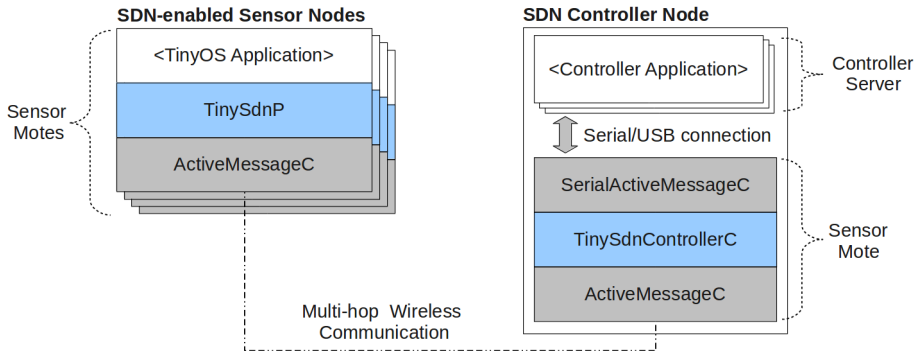


Figure 1. TinySDN Architecture Components [de Oliveira et al. 2014]

SDN-enabled sensor node is the component that runs on sensor motes, providing the forwading service to applications as a network protocol. As discussed

in [Luo et al. 2012], end devices are considered peripheral to SDN and hence out of the scope of OpenFlow, the main SDN project nowadays. On the other hand, sensor nodes behave like end devices by generating data packets to transmit sensed data, in addition to merely forwarding data as SDN switches do. Thus, *SDN-enabled sensor node* is a type of node that plays both roles: SDN switch and SDN end device.

SDN Controller Node is the node (or nodes in case of multiple controllers) that performs SDN controller tasks, i.e., executes the control plane: keep a network topology view, and apply definitions of SDN applications by creating and managing network flows using the specific protocol described in [de Oliveira et al. 2014]. Each *SDN-enabled sensor node* must find an *SDN controller node* to assign and then start to send network topology information and request flow specifications when necessary.

Next, we describe the specification of flows and actions protocol, the two types of nodes, as well the procedures to *SDN-enabled sensor nodes* find an *SDN controller node* and to establish communication with it, and to collect network topology information.

2.1. SDN-enabled Sensor Node

As shown in Figure 1 (left), *SDN-enabled sensor node* is composed of three parts: *ActiveMessageC*, *TinySdnP*, and *TinyOS Application*.

The *ActiveMessageC* is a TinyOS component that manages and provides programming interfaces to interact with the radio module of the sensor node, which comprehends the functions of link and physical layers. It is used by TinySDN to perform all tasks related to medium access and wireless communication, such as link quality estimation and control/data packet sending/receiving.

The *TinySdnP* is the main component of TinySDN, which is responsible for checking if a received packet matches a flow in the `flow table` and then perform the related action, or otherwise sends a flow request to an *SDN controller node*. Thus it is responsible for performing `flow table` update when receiving a flow setup response.

The *TinyOS Application* part is the equivalent to end-user device; it generates data packets and then places them on the network using the programming interface provided by the TinySDN component. It should be written by the network programmer, according to the WSN application, but we provide an example of this.

2.1.1. Finding a Controller and Establishing Communication with It

The first task of an *SDN-enabled Sensor Node* at network startup is to find an *SDN Controller Node* and establish communication with it in order to send control packets. For this, TinySDN employs the collection tree protocol (CTP) [Gnawali et al. 2009] as underlying protocol. In addition to being a widely used protocol in TinyOS-based applications, we selected CTP because of two features:

- **Hardware independence –** it adopts as metric the `ETX` value given by TinyOS Four-Bit Link Estimator Component [Fonseca et al. 2007], a software component that estimates the link quality between single-hop neighbors from the amount delivered or lost messages instead of using the traditional received signal strength indication (RSSI), a specific and hardware-dependent feature.

- **Multiple SDN controllers –** it uses a tree routing to deliver data through the network to a sink node, named root. Each node unconsciously adopts a root by sending data through the route with the lowest `ETX` sum. Thus, when multiple roots are announced each node adopts the root with best `ETX` without being aware that there is more than one of them, which builds a collection tree for each root.

2.1.2. Network Topology Information Collection

Network topology information collection is an important feature of TinySDN that enables the *SDN controller node* to build a view of the network by receiving information about sensor nodes and wireless links between them. It consists in each *SDN-enabled sensor node* recognizing its neighbors and measuring the quality of links with them, and then sending this information to the *SDN controller node* through the CTP route.

In order to recognize the neighborhood *SDN-enabled sensor nodes* uses the Four-Bit Link Estimator broadcast `beacon packet` and wait for responses. Upon receiving responses, each response sender is added to the `neighbor table` including the link quality metric (`ETX`). To allow us to obtain the neighbor table we needed to modify the Link Estimator interface and add a command as well.

Table 1 contains an example of `neighbor table`. The *Neighbor Id* column contains neighbors `TinyOS node id` (network node address) and the Link Quality column contains neighborhood measured link quality to correspondent neighbor. A lower `ETX` value means better quality, and infinity means that link was not sufficiently evaluated or the link is not available at the moment.

Table 1. Neighbor Table example.

Neighbor Id	*Link Quality (ETX)*
3	10
5	50
9	∞

2.1.3. Provided Programming Interfaces

The TinySdnP component provides to TinyOS application programmer two main interfaces:

- AMSend – it is an interface that enables sending data through the TinySDN tool protocol by calling the `send` command function, which receives three parameters: `data flow id` that identifies the flow to be used, a pointer to the `message packet` to be sent, and packet `length`. Furthermore, when instantiating this interface, it is mandatory to implement the `send done` event, which handles results (callbacks) of send command calls.
- Receive – it is an interface that enables receiving data through the TinySDN tool protocol by handling the packet reception event in a callback function implementation. This receive event implementation receives from the lower software lay-

ers the following parameters: `received packet`, a pointer to the packet's `payload`, and packet `length`.

2.2. SDN Controller Node

As observed in Figure 1 (right), it is composed of two different modules to be installed in different devices: sensor mote module and controller server module, described below.

- **Sensor mote module:** this module, which runs on a sensor mote, is responsible for communicating with *SDN-enabled sensor node* using *ActiveMessageC*. It uses *SerialActiveMessageC* to forward received messages from network to the *controller server module* and to forward messages from it to the network. The *TinySdnControllerC* part adapts messages and manages this communication.
- **Controller server module:** module that concentrates the control plane logic, i.e., keep a network topology view, hosts controller applications and apply definitions of it by creating and managing network flows. This module should run over a device with more resources that a sensor mote, such as a credit-card-sized computer running a Linux distribution, a laptop, a desktop or even a server. TinySDN defines the interfaces to be used in order to interact with the *sensor mote module*, and we provide a use case as a simple controller application, but more elaborated scenarios and applications could be defined by the network programmer.

2.3. Specification of Flows and Actions

A flow and its actions are a set of instructions used by the *SDN controller node* to program *SDN-enabled sensor nodes* behavior by sending flow setup packets that are translated into entries on `flow tables`. Thus, *SDN-enabled sensor nodes* are responsible for classifying packets in different flows and performing its corresponding actions.

TinySDN includes specification of two types of flows: control flow and data flow.

Data flows are used to forward data packet generated by applications through the sensor network. Three data flow actions are specified in TinySDN:

- Forward – classified data packet should be forwarded to a next hop sensor node specified in the action parameter field.
- Receive – classified data packet should be forwarded to application layer, i.e., delivered to the application.
- Drop – classified data packet should be dropped.

Data flows are specified as entries in the `data flow table`, where each entry is composed of four fields: *Flow ID*, field that identifies the flows and is used for packet classification; *Action*, which specifies the action to be performed; *Action Parameter*, field that contains a specific value related to action (e.g., in case of "Forward" action it specifies the next hop node id); and *Count*, field incremented when packets are matched to the flow entry, providing statistics. Table 2 (left) contains a `data flow table` example.

Control flows are used to forward control packets from the *SDN controller node* to *SDN-enabled sensor nodes*, i.e., it is just used to establish downward routes to control packets, since upward routes (from controller to sensor nodes) establishment is performed by CTP. Control flow entries have a unique and default action: forward; it is because only the *SDN controller node* generates control packets and the control flow identification is the node id (or address). A control flow table example is presented in Table 2 (right), it includes two fields: *destination node id* and *next hop node id*.

Table 2. Data Flow Table and Control Flow Table examples.

Flow ID	*Action*	*Action Parameter*	*Count*
2	Forward	5	2
9	Receive	-	5
11	Drop	-	2
34	Forward	10	50

Destination Node Id	*Next Hop Node Id*
5	5
10	5
13	5
15	3

3. Demo Proposal

The proposed demo will depict a WSN application that senses and transmits such data to a sink periodically (every 2 seconds) using TinySDN as communication protocol. Its purpose is to provide an overview of the TinySDN main features, focusing on the flow creation process and the use of such flows to deliver data from source sensor nodes to destination nodes. Therefore, the demo is designed to enable the visualization of all network events related to those two tasks.

To achieve the goal we use COOJA [Osterlind et al. 2006]. COOJA is a cross-level sensor network simulator that adopts a hybrid approach being able to simulate the network level through the environment implemented in Java, and to emulate the operating system level by running native sensor node program integrating it with simulated network environment using the Java Native Interface. Since COOJA runs cross-compiled binaries for the TelosB[2] mote by employng the TI MSP430 emulator in its environment, it is enabled to emulate/simulate TinyOS applications. By adopting an emulation/simulation environment, the demonstration is not susceptible to uncontrolled interference. Furthermore, COOJA provide a graphical user interface and features to capture and visualize mote output and network messages, such as beacon, control, and data packets.

In the demonstration, we adopt two topologies: grid topology (or mesh), which is very common in WSN application and illustrates a network with medium to high density; and string topology to illustrate a low-density network. The flows setup are executed reactively for both types, data flows and control flows, as depicted in Figure 2.

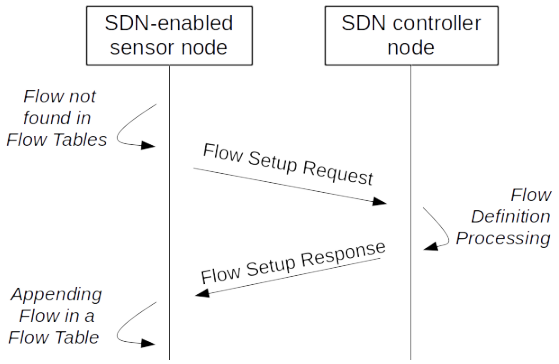


Figure 2. Reactive flow setup

[2]http://www.memsic.com/userfiles/files/Datasheets/WSN/6020-0094-02_B_TELOSB.pdf

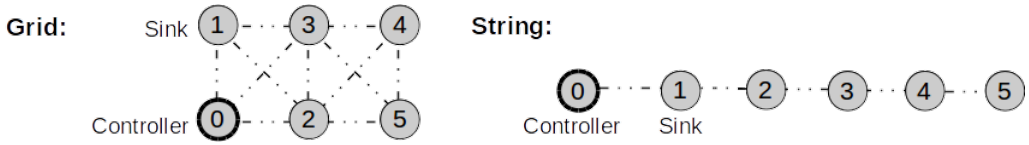


Figure 3. WSN topologies for demonstration

Once the control messages are exchanged, nodes will be able to send data to the sink. As depicted in Figure 3, the sensor nodes whose *node id* range from 1 to 5 are deployed with TinySDN, becoming *SDN-enabled sensor nodes*. In this demo they generate rules requests and perform the forwarding task according to the *Controller* instructions. Additionally, the nodes whose *node id* are 3, 4 or 5 transmit a counter (simulating data sensed). The node that *node id* $= 1$ is set as the data sink. A small number of nodes was selected in order to allow one to follow and understand the messages exchanged and displayed by COOJA. We could also increase the number of nodes to demonstrate TinySDN working with larger networks.

4. Used Resources, Source Code and Documentation Availability

TinySDN was implemented based on TinyOS[3] and its components, which we highlight: the collection tree protocol and the 4-bit Link Estimator. Thus, the TinyOS toolchain, including the build system, is a requirement to compile our tool.

We used TelosB mote as a platform during the development process and for the validation tests. In theory, TinySDN can run on any TinyOS-supported target hardware; however, additional tests should be performed for a reliable deployment based on other platforms. Also, we used COOJA to perform network environment tests.

TinySDN tool source code and the related documentation are available at `http://www.larc.usp.br/~cbmargi/TinySDN/`.

5. Considerations and Future Work

We introduced TinySDN, a tool designed to enable the TinyOS compatible platforms to Software-Defined Networking and allows the use of multiple controllers. Among the main benefits of using SDN paradigm in WSN scenarios, we highlight easier network protocol development and its hypothesis testing, as well as wireless sensor network management.

Currently, we are developing a controller with more functionalities in order to replace the current standalone controller. We also expect contributions from the community interested in improving the tool. Another future work topic is to extend matching rules by adding flexible bit masking capability, which enables interoperability with other protocols, but it implies a higher cost of communication and memory usage.

Acknowledgment

This work is funded by São Paulo Research Foundation (FAPESP) under grants #2013/15417-4 and #2014/06479-9. Cíntia Borges Margi is supported by CNPq research fellowship #307304/2015-9. The authors would like to thank the Laboratory of Computer Networks and Architecture of USP for providing the necessary infrastructure.

[3]Our implementation is based on the most recent released version of TinyOS (2.1.2), released on August 20, 2012. More information can be found at `http://www.tinyos.net`.

References

[Costanzo et al. 2012] Costanzo, S., Galluccio, L., Morabito, G., and Palazzo, S. (2012). Software defined wireless networks: Unbridling sdns. In *Proceedings of the 2012 European Workshop on Software Defined Networking*, EWSDN '12, pages 1–6, Washington, DC, USA. IEEE Computer Society.

[Culler et al. 2004] Culler, D., Estrin, D., and Srivastava, M. (2004). Overview of sensor networks. *Computer Magazine*, 37(8):41–49.

[de Oliveira et al. 2015] de Oliveira, B. T., Alves, R. C. A., and Margi, C. B. (2015). Software-defined wireless sensor networks and internet of things standardization synergism. In *Standards for Communications and Networking (CSCN), 2015 IEEE Conference on*, pages 60–65.

[de Oliveira et al. 2014] de Oliveira, B. T., Margi, C. B., and Gabriel, L. B. (2014). TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. In *Communications (LATINCOM), 2014 IEEE Latin-America Conference on*, pages 1–6.

[Fonseca et al. 2007] Fonseca, R., Gnawali, O., Jamieson, K., and Levis, P. (2007). Four bit wireless link estimation. In *Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets VI)*.

[Galluccio et al. 2015] Galluccio, L., Milardo, S., Morabito, G., and Palazzo, S. (2015). SDN-WISE: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 513–521.

[Gante et al. 2014] Gante, A. D., Aslan, M., and Matrawy, A. (2014). Smart wireless sensor network management based on software-defined networking. In *Communications (QBSC), 2014 27th Biennial Symposium on*, pages 71–75.

[Gnawali et al. 2009] Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., and Levis, P. (2009). Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14. ACM.

[Hill et al. 2000] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. (2000). System architecture directions for networked sensors. *SIGPLAN Notices*, 35(11):93–104.

[Luo et al. 2012] Luo, T., Tan, H.-P., and Quek, T. Q. S. (2012). Sensor openflow: Enabling software-defined wireless sensor networks. *IEEE Communications Letters*, 16(11):1896–1899.

[Mahmud and Rahmani 2011] Mahmud, A. and Rahmani, R. (2011). Exploitation of openflow in wireless sensor networks. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 1, pages 594–600.

[Osterlind et al. 2006] Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. (2006). Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648.

Salão de Ferramentas do SBRC 2016
Sessão Técnica 3
Computação em Nuvem

Cloud Capacitor: Uma Ferramenta de Apoio ao Planejamento da Capacidade de Aplicações na Nuvem

Matheus Cunha, Marcelo Gonçalves, Nabor C. Mendonça, Américo Sampaio

[1]Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321, Edson Queiroz, CEP 60811-905 Fortaleza, CE

{mathcunha,marcelocg}@gmail.com,{nabor,americo.sampaio}@unifor.br

Resumo. *Este trabalho apresenta* Cloud Capacitor*, uma ferramenta para apoiar o planejamento da capacidade de aplicações em nuvens que oferecem infraestrutura-como-serviço (IaaS). A ferramenta é baseada em uma nova abordagem para avaliar o desempenho de aplicações na nuvem, denominada* inferência de desempenho*. Essa abordagem tem como premissa a definição de uma relação de capacidade entre diferentes configurações de recursos de um dado provedor de nuvem IaaS, a partir da qual é possível prever (ou "inferir") o desempenho esperado de uma aplicação para uma ampla variedade de cenários de implantação, sendo que apenas uma pequena parte desses cenários precisa de fato ser implantada e executada na nuvem. A utilização da ferramenta é ilustrada através de um exemplo envolvendo a avaliação da aplicação de* blogging *WordPress no provedor de nuvem Amazon EC2.*

Abstract. *This work presents* Cloud Capacitor*, a tool to support application capacity planning in infrastructure-as-a-service (IaaS) clouds. The tool is based on a novel performance evaluation approach for cloud applications, named* performance inference*. This approach relies on the definition of a capacity relation between different resource configurations offered by a given IaaS cloud provider, enabling one to predict (or "infer") an application's expected performance for a large variety of deployment scenarios, with the benefit that only a small fraction of those scenarios need to be effectively deployed and executed in the cloud. The use of the tool is illustrated through an example evaluation of the WordPress blogging application in the Amazon EC2 cloud.*

1. Introdução

Um dos principais desafios enfrentados pelos usuários de nuvens que oferecem infraestrutura-como-serviço (IaaS) é planejar adequadamente a capacidade dos recursos da nuvem necessários para atender as demandas específicas de suas aplicações [Menascé and Ngo 2009]. Tipicamente, esse problema de planejamento tem sido atacado em duas frentes: através do uso de ferramentas de predição de desempenho na nuvem (por exemplo, [Li et al. 2011, Fittkau et al. 2012, Jung et al. 2013]), ou empiricamente, medindo-se o desempenho real da aplicação na nuvem quando implantada sob diferentes configurações de recursos e submetida a diferentes níveis de carga de trabalho — o que normalmente é feito com o auxílio de ferramentas automatizadas para avaliação de desempenho na nuvem (por exemplo, [Jayasinghe et al. 2012,

Silva et al. 2013, Cunha et al. 2013a]. Por executarem a aplicação no próprio ambiente de nuvem, ferramentas de avaliação de desempenho conseguem resultados muito mais precisos no que diz respeito à seleção das melhores configurações de recursos para demandas específicas. No entanto, uma limitação importante desses trabalhos é a necessidade de se testar exaustivamente a aplicação alvo sob uma grande quantidade de configurações de recursos e de cargas de trabalho, implicando em altos tempo e custo durante a fase de planejamento.

Este trabalho apresenta a ferramenta *Cloud Capacitor*, a qual faz uso de *inferência de desempenho* [Gonçalves et al. 2015a, Gonçalves et al. 2015b] para reduzir o número de configurações de recursos e de cargas de trabalho que precisam ser efetivamente avaliadas na nuvem, reduzindo, assim, o tempo e o custo da fase de planejamento. A abordagem de inferência de desempenho utilizada tem como premissa a definição de uma relação de capacidade entre diferentes configurações de recursos de um dado provedor de nuvem IaaS, a partir da qual é possível prever (ou "inferir"), com alta precisão, o desempenho esperado de uma aplicação para uma ampla variedade de cenários de implantação, sendo que apenas uma pequena parte desses cenários precisa de fato ser implantada e executada na nuvem [Gonçalves et al. 2015a, Gonçalves et al. 2015b].

A próxima seção descreve o projeto e a implementação da ferramenta *Cloud Capacitor* na forma de uma biblioteca para criação de sistemas de apoio ao planejamento de capacidade na nuvem. A Seção 3 apresenta um desses sistemas, *Capacito,Web*, que disponibiliza uma interface web para acesso aos serviços oferecidos pela biblioteca *Cloud Capacitor*. A Seção 4 compara a ferramenta proposta com outros trabalhos relacionados. Por fim, a Seção 5 oferece as conclusões e sugestões para trabalhos futuros.

2. *Cloud Capacitor*

Cloud Capacitor é uma biblioteca para criação de sistemas de avaliação e inferência de desempenho de aplicações em ambientes de nuvem IaaS, implementada como um *gem* (pacote) da linguagem Ruby. A documentação e o código fonte da biblioteca estão disponíveis em `https://github.com/marcelocg/cloud_capacitor`.

2.1. Classes e Responsabilidades

A biblioteca é composta por um conjunto de classes que representam entidades relevantes no domínio da avaliação da capacidade de aplicações em ambientes de nuvem (ver Figura 1). A classe principal, *Capacitor*, implementa o fluxo básico do processo de avaliação e inferência de desempenho, com todos os seus pontos de decisão e de extensão. Maiores detalhes sobre o funcionamento do processo podem ser obtidos em [Gonçalves et al. 2015a].

Entre os pontos de extensão, destaca-se o uso da classe *Strategy*, que pode ser especializada para a criação de diferentes estratégias de avaliação (por exemplo, estratégias com comportamento otimista, pessimista ou conservador em relação à escolha das configurações de recursos e das cargas de trabalho a serem avaliadas [Gonçalves et al. 2015a]). Outro ponto de extensão oferecido pela biblioteca é a classe *DefaultExecutor*, que pode ser especializada para utilizar diferentes ferramentas de implantação e execução de aplicações na nuvem (na Figura 1, a classe *DefaultExecutor* é especializada pela classe *RealExecutor*).

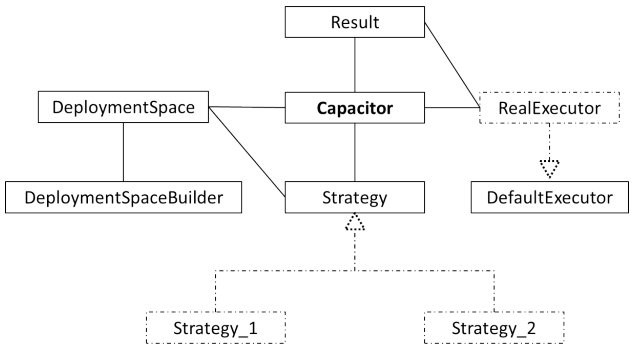


Figura 1. Diagrama de classes da biblioteca *Cloud Capacitor*.

Para que o processo de avaliação de capacidade possa ser efetuado, a classe *Capacitor* deve conhecer o resultado de cada execução da aplicação alvo, de modo que possa tomar as decisões corretas na indicação das configurações que atenderiam ou não os seus requisitos de desempenho, também conhecidos como *Service Level Objectives* (SLOs). No contexto deste trabalho, essas configuração são rotuladas como candidatas e rejeitadas, respectivamente. Os resultados de cada cenário de implantação avaliado são encapsulados na classe *Result*, cujos atributos são fornecidos pela subclasse responsável pela execução dos testes de desempenho da aplicação alvo na nuvem.

Finalmente, durante a execução do processo, a classe *Capacitor* precisa ter conhecimento das configurações de recursos disponibilizados pelo provedor de nuvem nas quais a aplicação alvo será implantada e avaliada, aqui referenciadas como *espaço de implantação*. Essa é a responsabilidade da classe *DeploymentSpace*, que implementa uma estrutura de dados em memória para representar os diversos níveis de capacidade definidos entre as configurações de recursos selecionadas para avaliação. A construção dessa estrutura é responsabilidade da classe *DeploymentSpaceBuilder*, que contém os algoritmos necessários à geração do grafo direcionado usado para navegação pelos níveis de capacidade definidos entre as configurações do espaço de implantação.

2.2. Fluxo de Utilização

A utilização da biblioteca na criação de um sistema de avaliação de capacidade deve seguir os seguintes passos:

1. Configurar os parâmetros de criação do espaço de implantação.
2. Identificar os tipos de máquinas virtuais do provedor de nuvem que irão compor o espaço de implantação.
3. Instanciar um objeto *Capacitor*.
4. Atribuir ao *Capacitor* um objeto *DefaultExecutor*.
5. Atribuir ao *Capacitor* um objeto *Strategy*.
6. Executar o método *run_for* do *Capacitor*.

A configuração do espaço de implantação é feita através de um arquivo em formato YAML[1] e inclui apenas dois parâmetros: (i) o custo máximo permitido para cada configuração (calculado pela soma dos custos de cada instância que compõe a

[1] `http://www.yaml.org/`

```
1  - !ruby/object:CloudCapacitor::VMType
2    name: m3.xlarge
3    cpu: 4
4    mem: 15
5    price: 0.28
6    category: m3
7  - !ruby/object:CloudCapacitor::VMType
8    name: c3.large
9    cpu: 2
10   mem: 3.75
11   price: 0.105
12   category: c3
```

(a)

```
1  cap = Capacitor.new
2  cap.executor = Executors::RealExecutor.new
3  cap.strategy = Strategies::Strategy.new
4
5  cap.strategy.approach workload::optimistic,
6                        conf::conservative
7
8  candidates = cap.run_for [100,200,300,400]
9
10 total_executions = cap.executions
11
12 total_cost = cap.run_cost
```

(b)

Figura 2. Exemplos de uso da biblioteca *Cloud Capacitor*: (a) especificação de tipos de máquinas virtuais; (b) código para execução do processo de avaliação.

configuração); e (ii) o número máximo de instâncias permitidas para cada configuração. Por exemplo, se os dois parâmetros acima foram configurados com os valores $7,00$ e 4, respectivamente, então serão criadas configurações com 1, 2, 3 e 4 instâncias para cada tipo de máquina virtual especificada, desde que o custo total da configuração não ultrapasse o valor de $7,00$ unidades monetárias.

O próximo passo na utilização da biblioteca é especificar os tipos de máquinas virtuais a serem utilizados na geração do espaço de implantação. Essa especificação é feita em outro arquivo em formato YAML, discriminando as características de CPU, memória e custo de cada tipo de máquina virtual. Essas informações serão utilizadas pela classe *DeploymentSpaceBuilder* para a criação das configurações de recursos que comporão o espaço de implantação, atendendo às restrições de tamanho e custo impostas no passo anterior, bem como para a definição de relações de capacidade entre elas. A Figura 2(a) mostra um exemplo da especificação em YMAL de dois tipos de máquinas virtuais (quais sejam, *m3.xlarge* e *c3.large*) oferecidos pelo provedor Amazon EC2.[2]

Definidas as configurações necessárias, o desenvolvedor pode, assim, criar um objeto a partir da classe *Capacitor* e, então, atribuir a ele um objeto instanciado a partir de uma subclasse de *DefaultExecutor*, cuja implementação deve fornecer os meios necessários para a implantação e execução dos testes de desempenho da aplicação alvo.

A Figura 2(b) mostra um exemplo de código em Ruby responsável pela criação e execução de um sistema de avaliação de capacidade implementado a partir da biblioteca *Cloud Capacitor*. Na linha 1, vê-se a instanciação do *Capacitor*. Na linha 2, um objeto da classe *RealExecutor* é atribuído ao *Capacitor*. A partir da linha 3, vêem-se os passos seguintes da utilização da biblioteca, com a definição de uma estratégia de avaliação, configurada com uma heurística de comportamento optimista para seleção de cargas de trabalho e conservador para seleção de configurações. A execução do processo de avaliação de capacidade acontece de fato a partir da chamada realizada na linha 8, onde são passados valores que representam uma lista de grandezas de demandas que serão impostas à aplicação alvo quando implantada e executa utilizando as configurações de recursos geradas como parte da construção do espaço de implantação.

Ao final da execução, o processo retorna a lista de configurações candidatas para cada valor de demanda passado como parâmetro. Adicionalmente, o desenvolvedor pode solicitar alguns dados a respeito da própria avaliação, como o número total de execuções da aplicação alvo na nuvem (linha 10) e o custo total dessas execuções (linha 12).

[2] `http://aws.amazon.com/ec2`

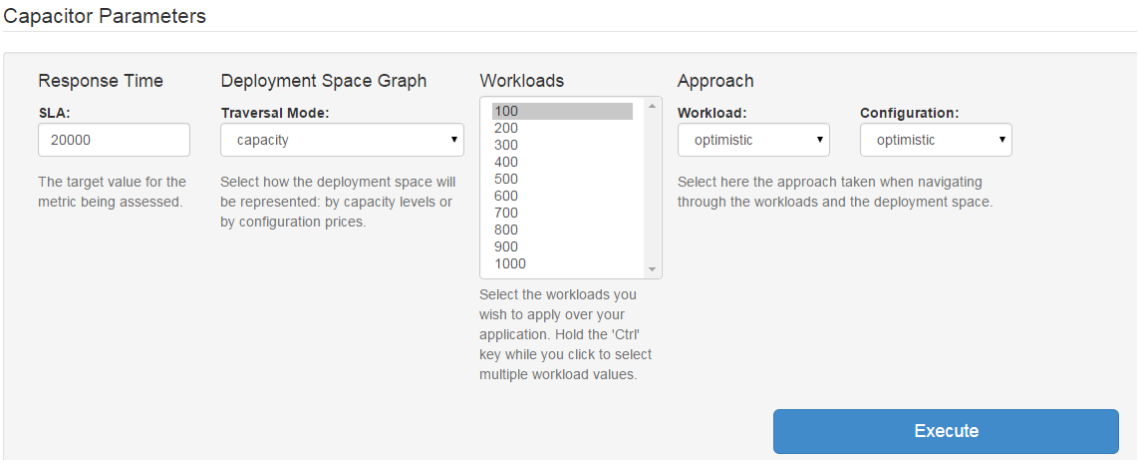


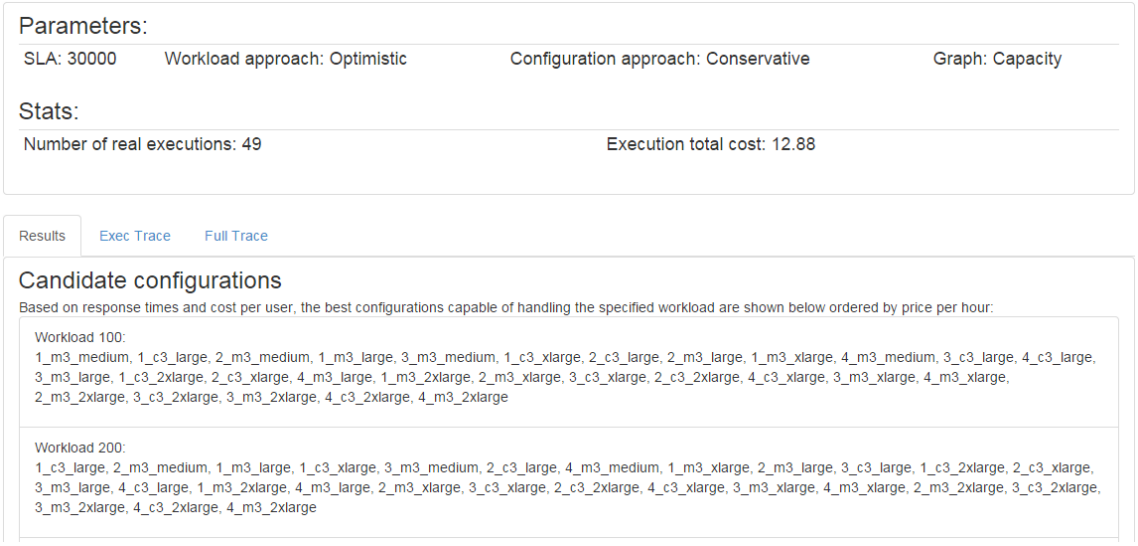
Figura 3. Página inicial do *Capacitor Web*.

3. *Capacitor Web*

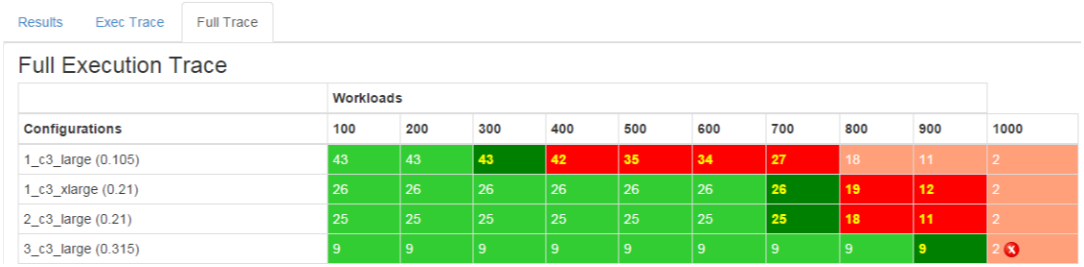
O *Capacitor Web* é uma aplicação web implementada utilizando o arcabouço *Ruby on Rails*[3] e que faz uso das funcionalidades oferecidas pela biblioteca *Cloud Capacitor* para a realização de testes de avaliação e inferência de desempenho em ambientes de nuvem. Essa aplicação foi utilizada para apoiar a condução de diversos experimentos de avaliação de capacidade envolvendo a aplicação de *blogging* WordPress[4] na nuvem Amazon EC2. Uma versão do *Capacitor Web*, pré-configurada com os resultados desses experimentos, está disponível em `https://capacitor-web.herokuapp.com/`.

A Figura 3 mostra a tela inicial da aplicação, onde o usuário tem a oportunidade de parametrizar a execução do processo de avaliação. O primeiro campo permite a especificação do SLO que deve ser respeitado pela aplicação para que uma execução seja considerada bem sucedida. Em seguida, apresenta-se o campo para seleção da representação das relações de capacidade entre as configurações do espaço de implantação, com opções de representação por capacidade (tipo de máquina virtual) ou por preço. Mais ao centro da tela, encontra-se uma lista de valores de carga, *Workloads*, essa lista permite que seja selecionada a demanda que será imposta à aplicação alvo e representa o número de usuários simultâneos utilizados nas execuções. Os últimos dois campos apresentam as opções a serem usadas na seleção da demanda e dos níveis de capacidade, respectivamente. As opções aqui são *Optimistic*, *Conservative* e *Pessimistic* para ambos os campos. A combinação desses dois parâmetros constitui a estratégia de avaliação que será utilizada pelo processo durante a exploração do espaço de implantação. Em linhas gerais, uma estratégia otimista seleciona os maiores níveis de demanda e os menores níveis de capacidade; já uma estratégia conservadora seleciona níveis intermediários tanto para a demanda quanto para a capacidade; por fim, uma estratégia pessimista seleciona os menores níveis de demanda e os maiores níveis de capacidade. A escolha da estratégia de avaliação é um ponto crucial da configuração do processo de planejamento de capacidade, uma vez que ela impacta diretamente na quantidade de cenários de implantação que serão efetivamente executados na nuvem [Gonçalves et al. 2015a].

[3] `http://rubyonrails.org/`
[4] `https://wordpress.org/`



(a)



(b)

Figura 4. Página de resultados do *Capacitor Web*: (a) dados de execução e configurações candidatas; (b) rastro com os resultados das avaliações.

Após a configuração dos parâmetros da página inicial, o usuário da aplicação pode solicitar a execução do processo acionando o botão "Execute", no canto inferior direito dessa mesma página. A Figura 4 mostra a página de resultados da aplicação, considerando dados obtidos a partir dos experimentos realizados com o WordPress na nuvem Amazon EC2. Essa página se divide em duas grandes áreas: uma superior, fixa, e uma inferior, que, por sua vez, se subdivide em três partes, separadas em abas distintas.

A parte superior mostra os dados passados como parâmetros pelo usuário na tela inicial, para facilitar a análise dos resultados e sua comparação futura. Assim, são exibidos o valor do SLO, as estratégias para seleção da demanda e dos níveis de capacidade das configurações, e a opção selecionada para a geração do espaço de implantação. Logo abaixo, são exibidos o número de execuções reais realizadas no ambiente de nuvem e o custo total (em US$ por hora) dessas execuções, com base no preço estabelecido pelo provedor para cada uma das configurações efetivamente utilizadas nos testes de desempenho da aplicação.

As três abas da parte inferior ativam a visualização das informações complementares resultantes da execução do processo. A primeira aba, com o título "Results", exibe a lista de configurações candidatas identificadas para cada nível de demanda avaliado (ver Figura 4(a)), conforme retornado pelo método *run_for* da classe *Capacitor*.

A segunda aba, com o título "Exec Trace", exibe os dados de rastreamento da

execução do processo. Esses dados permitem visualizar a sequência de execuções da aplicação alvo na nuvem, informando, para cada execução, o valor da demanda e a configuração de recursos utilizados.

Por fim, a terceira aba, com o título "Full Trace", exibe uma tabela cujas colunas e linhas representam, respectivamente, os diferentes valores de demanda e as diferentes configurações de recursos avaliados (ver Figura 4(b)). Dessa forma, cada célula da tabela representa um possível cenário de implantação da aplicação na nuvem, sendo que o número informado na célula indica a ordem em que aquele cenário foi avaliado e a sua cor e tonalidade indicam, respectivamente, o resultado dessa avaliação e o meio através do qual esse resultado foi obtido. Por exemplo, células verdes (vermelhas) indicam que a aplicação atingiu (não atingiu) o SLO desejado, enquanto cores em tons claros (escuros) indicam que os resultados foram obtidos por meio de inferência de desempenho ou de execuções reais da aplicação na nuvem. Dessa forma, uma maior presença de células em tons claros nessa aba indica uma maior redução no número de execuções da aplicação, implicando, portanto, em menores tempo e custo durante o processo de planejamento. Ainda nessa aba, uma célula contendo um círculo vermelho com a letra "x" indica um erro de inferência, ou seja, um resultado obtido via inferência de desempenho mas que não se confirmou na prática. Mais detalhes sobre a acurácia do processo de inferência de desempenho podem ser obtidos em [Gonçalves et al. 2015a].

4. Trabalhos Relacionados

As soluções existentes para avaliar a capacidade de aplicações em nuvens IaaS oferecem alta precisão, quando são baseadas em dados de desempenho obtidos diretamente da execução da aplicação alvo no provedor, como é caso das ferramentas *Expertus* [Jayasinghe et al. 2012], *CloudBench* [Silva et al. 2013] e *Cloud Crawler* [Cunha et al. 2013a, Cunha et al. 2013b]. Além disso, essas soluções oferecem grande flexibilidade de uso, no sentido em que permitem aos usuários avaliar diferentes combinações de componentes da aplicação sob as mais variadas configurações de recursos e demandas. O ponto negativo é a necessidade de executar cada uma das configurações definidas pelo usuário, uma vez que essas ferramentas não oferecem nenhum mecanismo voltado para reduzir a quantidade de execuções da aplicação. Dessa forma, cabe exclusivamente aos usuários dessas soluções definirem as melhores estratégias de explorar o espaço de implantação da aplicação na nuvem.

A ferramenta *Cloud Capacitor*, descrita neste trabalho, oferece uma alternativa mais eficiente às soluções de avaliação de capacidade de aplicações existentes, uma vez que reduz o número total de cenários sob os quais é necessário implantar e executar a aplicação na nuvem. Isso é possível com a utilização da abordagem de inferência de desempenho, também proposta pelos autores deste trabalho [Gonçalves et al. 2015a], a qual permite inferir, com alta precisão, o desempenho esperado de uma aplicação para uma ampla variedade de cenários de implantação, a partir da definição de relações de capacidade entre diferentes configurações de recursos de um dado provedor de nuvem.

5. Conclusão e Trabalhos Futuros

A tarefa de escolher adequadamente os recursos computacionais (em particular, máquinas virtuais) de um provedor de nuvem, de forma a minimizar os custos necessários para

atender diferentes níveis de demanda de uma aplicação, é um desafio importante para o qual ainda não existem soluções plenamente satisfatórias disponíveis. Este trabalho apresentou uma nova ferramenta de apoio ao processo de avaliação de capacidade de aplicações na nuvem, que tem como principal diferencial a utilização de uma abordagem de inferência de desempenho para reduzir o tempo e o custo tipicamente associados com esse tipo de atividade.

Com relação aos trabalhos futuros, algumas possibilidades interessantes para melhoria ou extensão da ferramenta proposta incluem: investigar novos critérios para a geração do espaço de implantação da aplicação, por exemplo, utilizando informações sobre o preço das configurações como fonte para a definição de suas relações de capacidade; implementar novas estratégias de avaliação, por exemplo, considerando informações sobre a utilização dos recursos da nuvem pela aplicação, como consumo de CPU e memória, diante da escolha dos níveis de capacidade e de demanda a serem avaliados; e realizar novos experimentos com outras aplicações e provedores de nuvem, visando não apenas validar a ferramenta mas também melhorá-la como produto.

Referências

Cunha, M. et al. (2013a). A Declarative Environment for Automatic Performance Evaluation in IaaS Clouds. In *IEEE CLOUD 2013*, pages 285–292.

Cunha, M. et al. (2013b). Cloud Crawler: Um Ambiente Programável para Avaliar o Desempenho de Aplicações em Nuvens de Infraestrutura. In *SBRC 2013*.

Fittkau, F. et al. (2012). CDOSim: Simulating cloud deployment options for software migration support. In *IEEE MESOCA 2012*, pages 37–46.

Gonçalves, M. et al. (2015a). Inferência de Desempenho: Uma Nova Abordagem para Planejar a Capacidade de Aplicações na Nuvem. In *SBRC 2015*. Prêmio de Melhor Artigo!

Gonçalves, M. et al. (2015b). Performance Inference: A Novel Approach for Planning the Capacity of IaaS Cloud Applications. In *Proc. of IEEE CLOUD 2015*, pages 813–820.

Jayasinghe, D. et al. (2012). Expertus: A Generator Approach to Automate Performance Testing in IaaS Clouds. In *IEEE CLOUD 2012*, pages 73–80.

Jung, G. et al. (2013). CloudAdvisor: A Recommendation-as-a-Service Platform for Cloud Configuration and Pricing. In *IEEE SERVICES 2013*, pages 456–463.

Li, A. et al. (2011). CloudProphet: Towards Application Performance Prediction in Cloud. In *ACM SIGCOMM 2011*, pages 426–427.

Menascé, D. A. and Ngo, P. (2009). Understanding Cloud Computing: Experimentation and Capacity Planning. In *CMG 2009*.

Silva, M. et al. (2013). CloudBench: Experiment Automation for Cloud Environments. In *IEEE IC2E 2013*, pages 302–311.

Rufus: Ferramenta para o gerenciamento de infraestrutura para a execução de aplicações em containers

Jonatan Souza[1,2], **Allan Santos**[1,2], **Matheus Bandini**[1], **Henrique Klôh**[1], **Bruno Schulze**[1]

[1] Laboratório Nacional de Computação Científica
(LNCC) – 25.651-075, Petrópolis, RJ, Brazil

[2]Faculdade de Educação Tecnológica do Estado do Rio de Janeiro
(FAETERJ-Petrópolis) – Petrópolis, RJ, Brazil

{jgsouza,amsantos}@faeterj-petropolis.edu.br, {mbandini,henrique,schulze}@lncc.br

Abstract. *This article presents Rufus, a tool to enable users who have little or no knowledge on infrastructure management to be able to submit jobs to be performed in a computation environment suitable for their application. This is achieved through the application submission and management in a cloud computing based on the container technology. With Rufus, one can create dedicated environments with specific configurations to meet the requirements of each application being executed. Authentication and management of users and files are also supported by Rufus.*

Resumo. *Este artigo apresenta a ferramenta Rufus, cujo objetivo é permitir que usuários que detenham pouco ou nenhum conhecimento sobre gestão de infraestrutura, sejam capazes de submeter trabalhos para serem executados em um ambiente computacional adequado para a sua aplicação. Isso se dá através da submissão e da gerência das aplicações em uma Nuvem Computacional baseada na tecnologia de Containers. Através da ferramenta, é possível criar um ambiente dedicado, com configurações personalizadas de acordo com as necessidades de cada aplicação a ser executada. Além disso, o Rufus oferece recursos como autenticação, gerência de usuários e gerência de arquivos.*

1. Introdução

O uso da computação científica permitiu o avanço dos demais campos da ciência com o desenvolvimento de pesquisas baseadas em simulações. Contudo, muitos centros de pesquisas ainda dependem da habilidade individual dos cientistas para o uso eficiente dos recursos computacionais necessários para realizar experimentos. Sem o auxílio de uma ferramenta que facilite a construção de *appliances* de aplicações e apoie a execução de *workflows*, o experimento pode ficar sujeito a falhas.

Um *workflow* científico pode ser definido como a descrição formal de um processo científico, dividido em passos, a serem executados em um determinado experimento [Deelman et al. 2009]. As aplicações científicas, que realizam cada um desses passos, necessitam, em geral, de ambientes configurados com dependências e ajustes específicos. As configurações necessárias para cada uma dessas aplicações podem ser completamente diferentes. Portanto, o compartilhamento de um mesmo ambiente por diversas aplicações pode gerar resultados indesejáveis para os gestores do ambiente.

A evolução da tecnologia de virtualização de recursos computacionais tornou-se extremamente útil, no tocante às propriedades dos *workflows* científicos, tendo em vista a complexidade de configuração das aplicações [Deelman et al. 2009] e dos recursos nos quais elas são executadas. O uso da virtualização permitiu o encapsulamento das aplicações em ambientes com recursos exclusivos, de forma a simplificar sua utilização voltada para sistemas distribuídos. Além disso, os ambientes virtualizados podem ser replicados na infraestrutura com o objetivo de aplicar o conceito DRY (*Don't Repeat Yourself*) e de facilitar a manutenção.

Recentemente, a virtualização em nível de sistema operacional é uma das formas de virtualização que mais cresce em popularidade. Isso, porque oferece um desempenho próximo do que é oferecido pelo *baremetal*. Como exemplo, pode-se citar o uso de *containers*, que é um método de virtualização que permite múltiplas instâncias de sistemas GNU/Linux executadas em um único *host*, enquanto dividem o mesmo *kernel* [Lamps et al. 2014]. Desta forma, a utilização de *containers* para a criação de *clusters* virtuais apresenta-se como uma alternativa efetiva e de baixo custo.

Este trabalho apresenta uma ferramenta, chamada Rufus, que foi desenvolvida visando aplicar o conceito de virtualização em nível de sistema operacional provendo *appliances* de aplicações através do LXC [1] (*LinuX Containers*) como infraestrutura para a execução de *workflows* científicos. Também é disponibilizada uma interface gráfica para a execução desses *workflows* em uma linguagem de alto nível com cliques e arrastes de *mouse*. Dessa forma, espera-se facilitar, para pesquisadores, as tarefas relacionadas ao gerenciamento e à replicação dos ambientes para a execução das aplicações. A ferramenta Rufus está dividida em dois níveis: Rufus-core e Rufus-web. O primeiro é responsável por realizar o gerenciamento dos recursos computacionais e o segundo é responsável pela interface gráfica e pela interação com o usuário.

A demonstração da ferramenta será realizada em duas etapas: a primeira envolve a criação de um *appliance* com as configurações necessárias para a execução de uma aplicação, enquanto a segunda consiste em submeter uma tarefa usando o *appliance* configurado através do módulo *Composer* do Rufus-web. A descrição completa dos procedimentos realizados na demonstração foram omitidas deste documento por limitação de espaço. No endereço `http://rufus.lncc.br/manual` é possível encontrar um tutorial passo-a-passo que guia o usuário através da instalação, configuração e uso da ferramenta. Na Seção 2 são apresentadas algumas ferramentas que se relacionam com o desenvolvimento ou o objetivo deste trabalho. A Seção 3 apresenta a ferramenta proposta. A Seção 4 apresenta a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

Ambientes utilizados para a execução de aplicações científicas são, em geral, compostos de máquinas com alta capacidade de processamento e de armazenamento, podendo haver o compartilhamento de tais recursos por diversas aplicações. Para evitar conflitos entre as diferentes aplicações e suas diferentes versões e permitir a utilização do ambiente de uma forma mais granular, é comum, segundo [Vogels 2008], isolar cada aplicação em um ambiente virtualizado. Esta seção descreve alguns aspectos de ferramentas que foram desenvolvidas com o objetivo de utilizar a virtualização para este fim.

[1] https://linuxcontainers.org/

O Juju [2] é um sistema orquestrador de serviços em nuvem desenvolvido pela Canonical Ltd. que também é a empresa que desenvolve o Ubuntu GNU/Linux. No contexto deste trabalho, as principais características do Juju são:

- O isolamento das aplicações pode ser feito através de máquinas virtuais ou *containers*;
- Um ambiente gráfico para a modelagem do ambiente;
- Ambiente multiusuário.

O Juju realiza o *deploy* de serviços em um ambiente de nuvem, além de permitir que as aplicações sejam isoladas em diferentes ambientes. Com o Juju é possível a utilização do OpenStack, do MaaS [3] (*Metal as a Service*) ou do LXC para o gerenciamento dos ambientes. A interface gráfica do sistema representa as aplicações com ícones em um *canvas* e as relações entre essas aplicações através de conexões que o usuário pode desenhar entre os ícones.

No Rufus, as conexões entre as aplicações representam a direção em que os dados gerados como resultado de uma podem ser passados como entrada para a próxima. As dependências entre as aplicações definem a ordem de execução, formando assim um *workflow*. O que se torna uma vantagem em relação ao Juju, pois este não oferece tal funcionalidade.

Docker [Boettiger 2015] e LXD [4] são duas ferramentas utilizadas para o gerenciamento de instâncias virtuais na forma de *containers* em um nível mais elevado de abstração do que o LXC. Ambas as ferramentas podem ser utilizadas através de API REST. Uma característica interessante do Docker é que ele é capaz de utilizar *containers* em sistemas operacionais diferentes de GNU/Linux. Além disso, no OpenStack também é possível utilizar o LXD [Short 2015] e o Docker [Fifield et al. 2014].

A solução utilizada neste trabalho para o gerenciamento de recursos procura ser mais simples do que as ferramentas citadas, no tocante a integração com os sistemas que possam se beneficiar do isolamento oferecido pela tecnologia de *containers*, como o portal de submissão e gerenciamento de aplicações descrito neste trabalho. Além de oferecer *containers* que, ao mesmo tempo, possam ser utilizados como uma instância da aplicação científica instalada, e permaneçam tão flexíveis quanto os criados diretamente pelo LXC.

Embora o uso de *containers* esteja amplamente difundido atualmente, muitas pesquisas científicas deixam de se beneficiar desta tecnologia, pois os seus experimentos são executados por alguma ferramenta de *workflow* científico sem algum suporte que automatize o gerenciamento de *containers*. O trabalho apresentado neste artigo propõe uma nova ferramenta de *workflow* científico que considera o uso de *containers* desde o início do seu desenvolvimento com o objetivo de obter a melhor integração possível entre as tecnologias.

Entretanto, existem também pesquisas sobre a adaptação de ferramentas de *workflow* científico existentes como [Zheng and Thain 2015] e [Gerlach et al. 2014]. O primeiro adapta as ferramentas Makeflow e WorkQueue para utilizar o Docker com o ob-

[2] http://www.ubuntu.com/cloud/juju

[3] http://www.ubuntu.com/cloud/maas

[4] https://linuxcontainers.org/lxd/

jetivo de encapsular as aplicações de *workflow*. Esta adaptação garante a reprodutibilidade dos experimentos eliminando as diferenças entre ambientes distintos que poderiam causar erros na execução de um *workflow*. O segundo apresenta o projeto Skyport, que também utiliza o Docker para adaptar o AWE/Shock com o objetivo de solucionar problemas de *deployment* e ampliar a eficiência da utilização de recursos.

3. Descrição da Ferramenta

A arquitetura da ferramenta descrita por este trabalho está divida em dois níveis: Rufus-web e Rufus-core, que podem ser observadas na Figura 1. Juntos, eles têm por finalidade oferecer as seguintes funcionalidades:

- Prover um controle de usuário divido por hierarquia;
- Proporcionar a gerência dos recursos computacionais através de uma interface gráfica;
- Permitir que o usuário crie e configure *appliances*. O *appliance* será considerado neste trabalho como uma instância que atenda as necessidades da aplicação;
- Prover um ambiente onde usuários possam submeter aplicações, utilizando *appliances* pré-configurados, através do próprio navegador.

Para oferecer essas funcionalidades, foi desenvolvido um portal *web* que somado a um motor gerador de *containers*, disponibiliza um ambiente completo ao pesquisador, com os objetivos de automatizar tarefas e facilitar a configuração da infraestrutura. O portal *web* provê uma interface para a gerência de recursos computacionais, autenticação de usuários, armazenamento de dados e a preparação e submissão de aplicações.

A gerência de infraestrutura se torna complexa a medida que a demanda por recursos computacionais aumenta. Além disso, a forma como esses recursos são oferecidos pode variar. Considerando essas características, as tarefas de gerenciamento da infraestrutura foram designadas a um componente independente, denominado Rufus-core. O Rufus-core é um motor gerador de *containers* e é responsável pela infraestrutura e por realizar as tarefas de gerência de recursos descritas no portal. Através dele, as aplicações são executadas.

3.1. Rufus-Core

O Rufus-core [Santos et al. 2015] é um motor gerador de ambientes virtualizados responsável pela criação de *containers* e por disponibilizá-los na forma de *appliances* pré-configurados dedicados às aplicações científicas. O objetivo é prover uma camada, acessível por *API REST*, a fim de atender aos requisitos das aplicações dos pesquisadores, criando dinamicamente ambientes para executar aplicações.

Os *containers* são criados através do LXC e as aplicações devem ser instaladas manualmente pelo administrador durante o processo de criação do *appliance*. Se o *container* oferecer o serviço *shell-in-a-box* [5], é possível requisitar uma URL ao Rufus-core para ter acesso a um terminal via navegador. Através deste, o administrador pode realizar remotamente qualquer instalação e configuração desejada no *container*.

O Rufus-core está dividido em dois módulos: i) `webservice` que implementa a *API REST* e realiza a comunicação com o portal Rufus-web; ii) `libpylxc` que realiza as

[5]https://code.google.com/p/shellinabox/

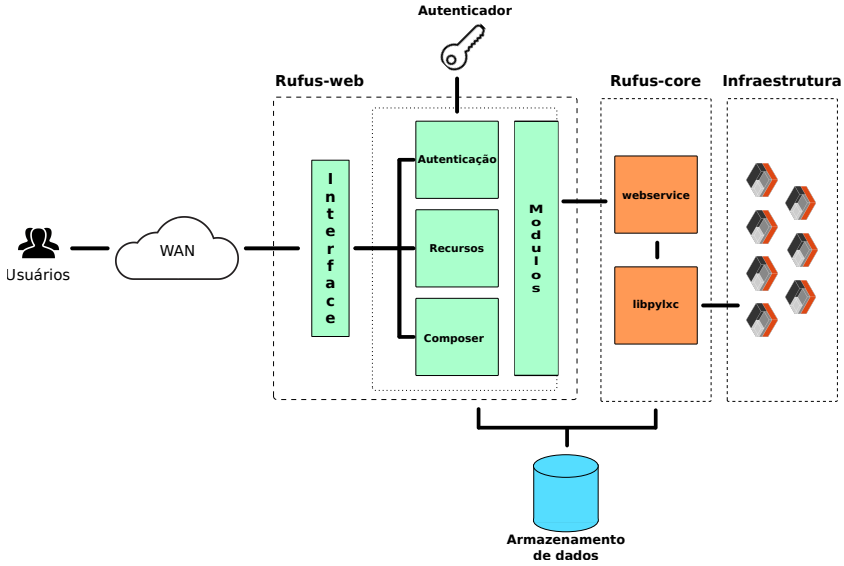


Figure 1. Visão geral da arquitetura da ferramenta Rufus.

operações necessárias para possibilitar as funcionalidades que o Rufus-core possui, como ilustrado na figura 1. O desenvolvimento desses módulos é feito em linguagem Python3, foi utilizado o *framework* Flask para o desenvolvimento do *webservice* e o formato JSON para a serialização das requisições feitas para a *API*. As funcionalidades disponíveis na API estão listadas a seguir:

- Listar, filtrar ou exibir em detalhes os *containers* existentes e também listar os *templates* e imagens do LXC.
- Mostrar recursos disponíveis na máquina real;
- Criar, clonar, alterar o estado e destruir *containers*;
- Disponibilizar *appliances* de aplicações científicas criados através de *containers* pré-configurados; e
- Executar aplicações em paralelo.

Para executar uma aplicação, é feito um clone do *container* no qual a aplicação foi instalada utilizando CoW (*Copy on Write*). Dessa forma, o desempenho do processo de clonagem não é prejudicado caso a aplicação seja muito grande, pois apenas referências aos arquivos originais são criadas. Os arquivos do *container* só são realmente copiados a medida que são modificados. Isso torna mais eficiente a criação de uma nova instância de uma aplicação. Para permitir o uso deste tipo de clonagem, foi utilizado o sistema de arquivos em camadas, chamado *OverlayFS* [6] que faz parte do *kernel Linux* nas versões recentes. Há também um módulo `client` que utiliza a biblioteca `requests` que foi desenvolvido com o objetivo de facilitar o uso da Rufus-core por *scripts* feitos em Python3, automatizando a criação das requisições HTTP.

3.2. Portal Rufus-web

O portal *web* Rufus tem o objetivo de prover ao usuário a interação com o sistema através de uma interface amigável. O portal está dividido em quatro módulos: **i)** Autenticação, **ii)** Gerência de Recursos, **iii)** Armazenamento de Dados e **iv)** Composer.

[6]https://www.kernel.org/doc/Documentation/filesystems/overlayfs.txt

Tendo em vista a segurança dos dados e também da rede na qual os recursos computacionais estão hospedados, foi implementado um módulo de autenticação baseado no sistema *OAuth2* [Hammer-Lahav and Hardt 2011]. Isso garante que apenas usuários autenticados no sistema poderão acessar as páginas do portal.

O Portal *web* divide os usuários em dois níveis:

- Administrador: usuários com esse privilégio têm acesso a todas as funcionalidades no que tange à gerência de recursos computacionais, sendo possível criar *appliances* e acessá-los via navegador através de um terminal *shell* e configurar as aplicações. Este tipo de usuário também conta com uma área na qual é possível tornar usuários comuns em administradores através do cadastro do e-mail;
- Pesquisador: este tipo de usuário é capaz apenas de elaborar e executar aplicações utilizando os *appliances* predefinidos pelos administradores.

O objetivo de apenas administradores terem acesso direto aos recursos computacionais é manter um número restrito de usuários com acesso direto aos *appliances* criados, aumentando assim a segurança do portal.

Com o objetivo de ampliar o controle sobre a infra-estrutura foi desenvolvido um módulo para a gerência de recursos. Neste módulo, através de uma interface gráfica intuitiva, administradores podem criar um *appliance* com o nome desejado, com um sistema *GNU/Linux* com diversas distribuições providas pelo Rufus-core. O módulo ainda provê um acesso remoto através de um terminal *shell* emulado no próprio navegador, através do qual é possível configurá-lo com as diretivas desejadas para a aplicação. Todos os *appliances* configurados pelo administrador estarão disponíveis para uso na área de submissão de *workflows*.

Para exportar os dados armazenados foi utilizado o NFS (*Network File System*) que é um sistema de arquivos distribuídos, a fim de compartilhar arquivos e diretórios entre computadores conectados em rede, formando um diretório virtual. Sendo assim, fica disponível ao Rufus-core as informações necessárias para a execução das tarefas.

No primeiro acesso ao portal Rufus, é criado um diretório com um identificador exclusivo para cada usuário. O e-mail obtido pelo autenticador é usado para nomear o diretório, tornado-o único e rastreável. Essa estrutura é importante, pois garante que cada usuário terá acesso apenas ao seu diretório. Ao fazer *upload* de um arquivo no portal, estes são salvos no subdiretório `files`. Após submetido um *workflow*, é criado um subdiretório com o nome do *workflow*. O Rufus-core, após a execução de cada *workflow* submetido, armazena os resultados. Esses arquivos podem ser obtidos através de *download*.

O módulo Composer permite ao pesquisador elaborar um *workflow* em um quadro de desenhos utilizando uma linguagem de alto nível com cliques e arrastes de *mouse*. Todas as instâncias configuradas pelos administradores ficam disponíveis na forma de *appliances* para o pesquisador no menu lateral. Cada *workflow* ao ser submetido deve ter no mínimo um arquivo e um *appliance*.

Quando um *workflow* é submetido, a requisição vai para o controlador de *workflows* do portal, que é responsável pela preparação para a submissão. O primeiro passo é organizar as informações dividindo em níveis (Figura 2), considerando suas dependências.

Cada nível então terá uma quantidade específica de aplicações a serem processadas, formando assim conjuntos de tarefas. Uma vez organizado, o controlador envia esse conjunto de tarefas para o Rufus-core de forma paralela. O controlador, então, aguarda o fim da execução deste conjunto de tarefas. Uma vez executado, o próximo conjunto é enviado para o gerente de recursos. Este ciclo se repete até alcançar o ultimo nível. Caso ocorra algum erro durante este processo o *workflow* é interrompido.

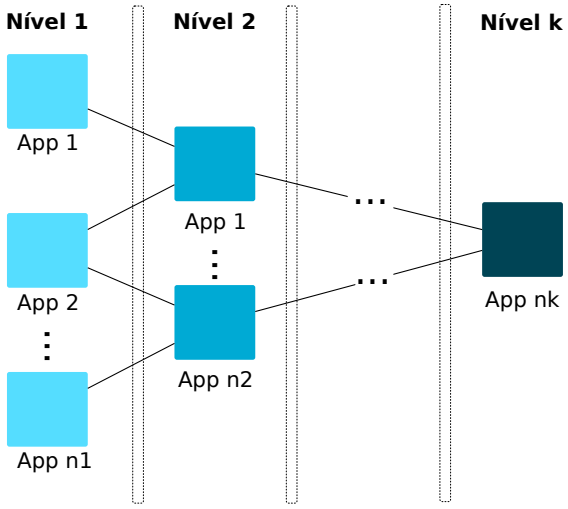


Figure 2. Exemplo de um *workflow* com K níveis

4. Conclusão e Trabalhos Futuros

O Portal *Rufus-web* permite a configuração de *appliances* e gerência de recursos computacionais, facilitando a submissão de *workflows* através de uma interface gráfica, considerando que as mesmas seriam tarefas onerosas quando realizadas sem o auxílio de uma ferramenta. A área de submissão de aplicações, (*Composer*), através do qual o pesquisador pode elaborar seu *workflow* ou aplicação e submetê-lo, é o grande diferencial e provê significativa contribuição à comunidade científica, tendo em vista a simplificação para a submissão dessas tarefas utilizando os *appliances* configurados com as aplicações.

Atualmente, o portal conta apenas com um gerente de recursos, denominado *Rufus-core*. Com a diversidade de infraestruturas de nuvens disponíveis, como por exemplo o *OpenStack* [Sefraoui et al. 2012] e o *OpenNebula* [Milojičić et al. 2011], a ideia é que o portal *Rufus-Web* proporcione uma gerência de *templates* capaz de integrá-las. Isso será possível com a implementação de interfaces que atendam às especificações de cada uma delas. Uma vez integradas, também será necessário a criação de uma inteligência capaz de escalonar a tarefa submetida, com o propósito de executá-la na infraestrutura que melhor se encaixe no perfil da aplicação, considerando que o portal comportará mais de um gerente de recursos.

Para que seja possível essa modularização, o Portal *Rufus-web* será divido em duas partes, sendo que a primeira possui o objetivo de permitir a interação e a autenticação de usuários, e uma segunda denominada *Rufus-AOD(AppFlow Organizer and Dispatcher)*. Esta última é responsável pela implementação das particularidades de cada gerente de recursos, além de realizar as requisições para execução das aplicações.

References

Boettiger, C. (2015). An introduction to docker for reproducible research. *SIGOPS Oper. Syst. Rev.*, 49(1):71–79.

Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2009). Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540.

Fifield, T., Fleming, D., Gentle, A., Hochstein, L., Proulx, J., Toews, E., and Topjian, J. (2014). *OpenStack Operations Guide*, chapter 4. "O'Reilly Media, Inc.".

Gerlach, W., Tang, W., Keegan, K., Harrison, T., Wilke, A., Bischof, J., D'Souza, M., Devoid, S., Murphy-Olson, D., Desai, N., et al. (2014). Skyport: container-based execution environment management for multi-cloud scientific workflows. In *Proceedings of the 5th International Workshop on Data-Intensive Computing in the Clouds*, pages 25–32. IEEE Press.

Hammer-Lahav, D. and Hardt, D. (2011). The oauth2. 0 authorization protocol. Technical report, IETF Internet Draft.

Lamps, J., Nicol, D. M., and Caesar, M. (2014). Timekeeper: a lightweight virtual time system for linux. In *Proceedings of the 2nd ACM SIGSIM/PADS conference on Principles of advanced discrete simulation*, pages 179–186. ACM.

Milojičić, D., Llorente, I. M., and Montero, R. S. (2011). Opennebula: A cloud management tool. *IEEE Internet Computing*, (2):11–14.

Santos, A. M. M., Kloh, H. M., Barbosa, J. P., and Schulze, B. (2015). Gerenciamento do motor gerador de containers para nuvens computacionais. In *ANAIS do XVI Workshop de Iniciação Científica em Arquitetura de Computadores e Computação de Alto Desempenho (WSCAD-WIC 2015)*. WSCAD-WIC.

Sefraoui, O., Aissaoui, M., and Eleuldj, M. (2012). Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3):38–42.

Short, C. (2015). Introduction to nova-compute-lxd. `https://insights.ubuntu.com/2015/05/06/introduction-to-nova-compute-lxd/`. [Online; accessed 01-February-2016].

Vogels, W. (2008). Beyond server consolidation. *Queue Virtualization Magazine*, 6(1):20–26.

Zheng, C. and Thain, D. (2015). Integrating containers into workflows: A case study using makeflow, work queue, and docker. In *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing*, VTDC '15, pages 31–38, New York, NY, USA. ACM.

REALIZAÇÃO

computação UFBA

UFBA 70 ANOS

INSTITUTO FEDERAL BAHIA

PROMOÇÃO

LARC

SBC Sociedade Brasileira de Computação

FOMENTO

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico

CAPES

APOIO

STI 40 Superintendência de Tecnologia da Informação | UFBA

POP-BA

PRONEX Filosofia e Ciências

FUNDAÇÃO Pierre Verger

RNP REDE NACIONAL DE ENSINO E PESQUISA

PATROCÍNIO DIAMANTE

nic.br Núcleo de Informação e Coordenação do Ponto BR

cgi.br Comitê Gestor da Internet no Brasil

PATROCÍNIO OURO

acm sigcomm

L3Networks

SOPHOS · arcserve

LogMeIn · KASPERSKY

GRUPO BINÁRIO Integração e Serviços

aruba a Hewlett Packard Enterprise company

PATROCÍNIO BRONZE

Google

IMTECH

Solutis tecnologias

PROTELE engenharia