SBRC 2015

33º SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS

18 a 22 de maio - Vitória - ES

ANAIS SBRC 2015

XXXIII Simpósio Brasileiro de Redes de Computadores

e Sistemas Distribuídos

18 a 22 de Maio de 2015

Vitória - ES

Anais
Trilha Principal do SBRC 2015

Coordenação do Comitê de Programa

Jussara Almeida (UFMG)
Christian Esteve Rothenberg (UNICAMP)

Coordenação geral

Magnos Martinello (UFES)
Moisés R. N. Ribeiro (UFES)

Realização

Universidade Federal do Espírito Santo (UFES)

Promoção

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

Produção Visual: Ateliê Eventos (Vitória-ES)

Produção Editorial: Celso Alberto Saibel Santos (UFES)

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 (33: 2015: Vitória, ES)

Anais do SBRC 2015, Dinâmicas, Sociais e Orientadas a Conteúdo / 33º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos; organizado por Jussara M. Almeida e Christian E. Rothenberg - Porto Alegre: SBC, 2015

808 p.

SBRC 2015 Realização: Universidade Federal do Espírito Santo

ISSN: 2177-496X

1. Redes de Computadores - Congressos. 2. Sistemas Distribuídos- Congressos.

Mensagem dos Coordenadores Gerais do SBRC 2015

Boas-vindas a todos os participantes do 33º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2015). Através dos anos, o SBRC vem se consolidando com um dos principais eventos científicos da área de Informática no país, trazendo sempre conceitos e práticas inovadoras nas áreas de Redes de Computadores e Sistemas Distribuídos. Nos sentimos extremamente honrados pela confiança em nós depositada para realizar esse importante evento pela primeira vez em Vitória, Espírito Santo.

O SBRC se caracteriza por proporcionar uma rica troca de ideias e experiências entre professores, pesquisadores, profissionais e estudantes atuantes na área de interesse do simpósio. É sempre um desafio manter o SBRC nos padrões de qualidade que o tem caracterizado em suas exitosas edições passadas.

A programação do SBRC 2015 engloba um conjunto de atividades bastante abrangente e de alta qualidade técnica. São 18 sessões técnicas nas quais serão apresentados 58 artigos completos, selecionados por meio de um rigoroso trabalho de revisão, envolvendo uma grande variedade de temas pertinentes e atuais relacionados às áreas de Redes de Computadores e Sistemas Distribuídos. A programação inclui ainda seis palestras proferidas por pesquisadores internacionalmente renomados e três painéis abordando temas extremamente atuais. São oferecidos seis minicursos, voltados à formação e atualização dos participantes sobre tópicos selecionados através de chamada pública. Adicionalmente, oito workshops são realizados em paralelo ao SBRC, focados em temas específicos e emergentes relacionados à área de interesse do simpósio. Nesta edição, também homenageamos com o prêmio Destaque SBRC uma personalidade das áreas de Redes de Computadores e Sistemas Distribuídos por sua contribuição significativa para a evolução da pesquisa e para a estruturação de uma sólida comunidade científica no Brasil.

As excelentes atividades programadas nesta edição são produto dos esforços de seus respectivos coordenadores. Um agradecimento muito especial a Jussara Almeida, Christian Esteve Rothenberg, Marinho Barcellos, Antônio Abelém, Antônio Augusto de Aragão, Sidney Lucena, Cesar Augusto Marcondes e Alfredo Goldman pelo tempo e empenho na efetivação das várias trilhas do SBRC 2015.

Ressaltamos o trabalho intenso e extremamente competente realizado pelo integrantes do Comitê de Organização Local ao qual gostaríamos de agradecer especialmente ao Rodolfo Villaça, Renato Moraes, Maxwell Monteiro, Maria José Pontes, Sabrina Felix, Marcia Paiva, Roberta Lima Gomes e Celso Alberto Saibel Santos. Agradecemos ainda às diretorias da SBC e do LARC, promotores do SBRC, pela confiança 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 prestado à organização do SBRC 2015.

Gostaríamos de agradecer ainda aos patrocinadores do Simpósio: o Comitê Gestor da Internet no Brasil, às agências governamentais de fomento – CNPq, CAPES e FAPES – e às empresas patrocinadoras por reconhecerem o SBRC como um evento importante para o fomento à pesquisa e inovação nas áreas de Redes de Computadores e Sistemas Distribuídos.

Por fim, agradecemos aos Departamentos de Informática e Engenharia Elétrica da UFES, e aos Programas de Pós-Graduação em Informática (PPGI) e Elétrica (PPGEE) por prestarem o indispensável suporte para a realização do SBRC.

Desejamos a todos os participantes que tenham uma excelente estadia em Vitória e que tirem proveito de todo o conhecimento que as atividades o SBRC 2015 tem a lhes oferecer.

Vitória, maio de 2015.

Magnos Martinello (*UFES*)
Moisés Renato Nunes Ribeiro (*UFES*)
Coordenadores Gerais

Mensagem dos Coordenadores do Comitê de Programa do SBRC 2015

Em mais de três décadas de realização, o SBRC já se consolidou como o mais importante evento científico nacional em Redes de Computadores e Sistemas Distribuídos e um dos mais concorridos na área de Ciência da Computação. Nesta 33ª edição, realizada em Vitória, a comunidade continuou a prestigiar o evento. Foram submetidos 186 artigos completos, dos quais 58 foram aceitos para apresentação e publicação, com uma taxa de aceitação em torno de 30%.

Os 186 artigos foram submetidos a uma avaliação criteriosa executada pelos 110 membros do Comitê de Programa e por 94 revisores associados a eles. Cada artigo foi avaliado por, pelo menos, 3 especialistas na área, sendo que vasta maioria dos artigos recebeu 4 pareceres. Após a etapa inicial de avaliação pelos revisores, os autores tiveram a oportunidade de submeter uma réplica (*rebuttal*) aos revisores, prestando os devidos esclarecimentos. Na sequência, os revisores tiveram acesso aos *rebuttals* dos artigos revisados e foram estimulados a discutir as avaliações com os demais co-revisores de cada artigo, com o objetivo de refletir sobre os seus pareceres e tentar reduzir eventuais discrepâncias nas avaliações, culminando com recomendações mais coerentes sobre uma potencial aceitação ou rejeição do artigo. Para todos os artigos que estavam na zona cinza (i.e., aqueles nem claramente aceitáveis e nem claramente rejeitáveis), convidou-se, então, mais um (em alguns casos dois) membro do Comitê de Programa, especialista no assunto, para avaliar os pareceres e dar uma opinião final sobre o destino de cada artigo. Finalizando todo o processo de seleção, os resultados foram consolidados, discutidos e homologados pelos membros do Comitê de Programa. Posteriormente, uma comissão aberta a todos os membros do Comitê de Programa foi organizada para escolha dos trabalhos destaque desta edição do evento.

Nos anais encontram-se os textos dos artigos completos selecionados, cobrindo tópicos bastante atuais como redes definidas por software, redes orientadas a conteúdo, computação em nuvem, tolerância a falhas e resiliência, e redes sem fio. Acreditamos que esses tópicos refletem bem a diversidade e o vigor das iniciativas de pesquisa em curso nas universidades, centros de pesquisa e empresas do País.

Como forma de promover uma maior visibilidade internacional dos excelentes trabalhos conduzidos pela comunidade, os autores dos melhores artigos serão convidados a submeter uma versão estendida de seus trabalhos para o *Journal of Internet Services and Applications* (JISA), da *Springer*, ou para a Revista Brasileira de Redes de Computadores e Sistemas Distribuídos (RB-RESD). Além disso, como na edição de 2014, todos os artigos aceitos (e apresentados) na Trilha Principal do SBRC 2015 serão publicados no IEEE Xplore, visando aumentar a visibilidade dos artigos junto à comunidade científica.

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. Ficamos muito satisfeitos pois sabemos que é somente com esse processo cuidadoso de avaliação e seleção de artigos que o alto nível de qualidade do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos pode ser mantido. Fazemos um agradecimento especial também aos coordenadores gerais deste 33° SBRC – professores Magnos Martinello e Moisés R. N. Ribeiro – pela confiança depositada em nosso trabalho e pelo apoio durante todo o processo. Também gostaríamos de agradecer a equipe de suporte do JEMS, que atendeu prontamente os nossos pedidos. Por fim, estendemos nossos agradecimentos aos autores que submeteram artigos ao SBRC 2015 pelo interesse no evento e por serem peça fundamental para seu sucesso.

Desejamos a todos os participantes do SBRC 2015 uma semana bastante produtiva e rica em discussões técnicas instigantes bem como conversas inspiradoras para novos projetos e cooperações. Aproveitem a sua estadia na linda cidade de Vitória!

Vitória, maio de 2015.

Jussara M. Almeida (UFMG)
Christian Esteve Rothenberg (UNICAMP)
Coordenadores do Comitê de Programa

Coordenadores do Comitê de Programa

Jussara M. Almeida (UFMG)
Christian Esteve Rothenberg (UNICAMP)

Membros do Comitê de Programa

Aldri dos Santos (UFPR)
Alex Vieira (UFJF)
Alfredo Goldman (IME-USP)
Ana Paula Couto da Silva (UFMG)
André Drummond (UnB)
André Soares (UFPI)
Anelise Munaretto (UTFPR)
Antônio Abelém (UFPA)
Antonio Alfredo Ferreira Loureiro (UFMG)
Antonio Rocha (UFF)
Antonio Tadeu Azevedo Gomes (LNCC)
Artur Ziviani (LNCC)
Bruno Schulze (LNCC)
Carlos Ferraz (UFPE)
Carlos Kamienski (UFABC)
Carlos Alberto Vieira Campos (UNIRIO)
Cesar Marcondes (UFSCAR)
Christian Esteve Rothenberg (UNICAMP)
Cristiano Both (UNISC)
Daniel Batista(IME-USP)
Daniel Fernandes Macedo (UFMG)
Daniel Guidoni (UFSJ)
Daniel Menasche (UFRJ)
Dênio Mariz Sousa (IFPB)
Djamel Fawzi Hadj Sadok (UFPE)
Dorgival Guedes (UFMG)
Edmundo Madeira (UNICAMP)
Eduardo Cerqueira (UFPA)
Eduardo Nakamura (UFAM)
Elias Duarte Jr. (UFPR)
Fabíola Greve (UFBA)
Fabio Kon (IME-USP)
Fatima Duarte-Figueiredo (PUC Minas)
Fábio Luciano Verdi (UFSCAR)
Fernando Dotti (PUC-RS)
Fernando Pedone (University of Lugano, Suiça)
Fernando Ramos (University of Lisbon, Portugal)
Flavia Delicato (UFRJ)
Francisco Brasileiro (UFCG)
Geraldo Robson Mateus (UFMG)
Gustavo Figueiredo (UFBA)
Gustavo Pavani (UFABC)
Horácio Oliveira (UFAM)
Humberto Marques (PUC-Minas)
Igor Moraes (UFF)

Italo Cunha (UFMG)
Jacir Bordim (UnB)
Jô Ueyama (USP)
Joni da Silva Fraga (UFSC)
José De Souza (UFC)
José Ferreira de Rezende (UFRJ)
José Augusto Suruagy Monteiro (UFPE)
Jose-Marcos Nogueira (UFMG)
Jussara Almeida (UFMG)
Katia Obraczka (University of California Santa Cruz)
Kelvin Dias (UFPE)
Kleber Cardoso (UFG)
Lau Cheuk Lung (UFSC)
Leandro Villas (UNICAMP)
Liane Margarida Rockenbach Tarouco (UFRGS)
Lisandro Zambenedetti Granville (UFRGS)
Luci Pirmez (UFRJ)
Luciano Paschoal Gaspary (UFRGS)
Luis Carlos Erpen de Bona (UFPR)
Luis Rodrigues (INESC-ID/IST)
Luis Henrique Costa (UFRJ)
Luiz Eduardo Buzato (UNICAMP)
Luiz Fernando Bittencourt (UNICAMP)
Luiz Fernando Soares (PUC-Rio)
Luiz Filipe Vieira (UFMG)
Magnos Martinello (UFES)
Marcel William Rocha da Silva (UFRRJ)
Marcelo Dias de Amorim (LIP6/CNRS and UPMC Sorbonne Universités, France)
Marcelo Rubinstein (UERJ)
Marco Spohn (UFFS)
Marcos Salvador (Lenovo)
Marcos Vieira (UFMG)
Marinho Barcellos (UFRGS)
Mauro Fonseca (PUC-PR)
Michael Stanton (RNP and UFF)
Michele Nogueira (UFPR)
Miguel Elias Mitre Campista (UFRJ)
Moisés R. N. Ribeiro (UFES)
Nabor Mendonca (UNIFOR)
Natalia Castro Fernandes (UFF)
Nazareno Andrade (UFCG)
Nelson Fonseca (UNICAMP)
Noemi Rodriguez (PUC-Rio)
Otto Carlos Muniz Bandeira Duarte (UFRJ)
Paulo Aguiar (UFRJ)
Paulo Cunha (UFPE)
Paulo Pires (UFRJ)
Paulo André da Silva Gonçalves (UFPE)
Pedro Velloso (UFF)
Rafael Pasquini (UFU)

Raimundo Jose de Araujo Macedo (UFBA)
Raquel Mini (PUC-Minas)
Rodolfo Villaça (UFES)
Rodrigo Fonseca (Brown University)
Ronaldo Ferreira (UFMS)
Ronaldo Salles (IME)
Rossana Andrade (UFC)
Sidney Lucena (UNIRIO)
Silvana Rossetto (UFRJ)
Stenio Fernandes (UFPE)
Thais Vasconcelos Batista (UFRN)
Vinod Rebello (UFF)
Wagner Meira Jr. (UFMG)
Weverton Cordeiro (Instituto Federal do Pará)
William Giozza (UnB)

Revisores

Adriano Prates (UFF)
Aldelir Fernando Luiz (Instituto Federal Catarinense)
Aldri dos Santos (UFPR)
Alex Pinho Magalhães (UFSC)
Alex Vieira (UFJF)
Alextian Liberato (IFES)
Alfredo Goldman (IME-USP)
Allan Vidal (Universidade Federal de São Carlos)
Alyson de Jesus dos Santos (COPPE/UFRJ)
Americo Sampaio (Universidade de Fortaleza)
Ana Paula Couto da Silva (UFMG)
André Drummond (Universidade de Brasília)
André Soares (Universidade Federal do Piauí)
Anelise Munaretto (UTFPR)
Antônio Abelém (Universidade Federal do Pará)
Antonio Alfredo Ferreira Loureiro (UFMG)
Antonio Rocha (IC/UFF)
Antonio Augusto Ribeiro Coutinho (Universidade Estadual de Feira de Santana)
Antonio Tadeu Azevedo Gomes (LNCC)
Arlindo Conceição (UNIFESP)
Artur Ziviani (LNCC)
Atslands Rocha (Universidade Federal do Ceará)
Bruno Costa (Universidade Federal do Rio de Janeiro)
Bruno Nunes (UFRJ)
Bruno Schulze (LNCC)
Bruno Silvestre (UFG)
Carlos Ferraz (UFPE)
Carlos Kamienski (UFABC)
Carlos Senna (Unicamp)
Carlos Alberto Vieira Campos (UNIRIO)
Carlos Henrique Nicodemus (Universidade Federal Fluminense)
Carmelo Bastos-Filho (University of Pernambuco)
Cesar Marcondes (Universidade Federal de São Carlos)

Claudio de Farias (Universidade Federal do Rio de Janeiro)
Cristian Machado (UFRGS)
Cristiano Both (UFRGS)
Dalbert Mascarenhas (CEFET-RJ)
Daniel Batista (IME – USP)
Daniel Dias (Universidade Federal do Rio de Janeiro)
Daniel Fernandes Macedo (UFMG)
Daniel Guidoni (UFSJ)
Daniel Lago (UNICAMP)
Daniel Menasche (UFRJ)
Darli Mello (UNICAMP)
Dênio Mariz Sousa (IFPB)
Denis Rosário (Federal University of Pará)
Dianne Medeiros (Universidade Federal do Rio de Janeiro)
Diego Kreutz (Faculty of Science of University of Lisbon)
Diogo Mattos (Universidade Federal do Rio de Janeiro)
Djamel Fawzi Hadj Sadok (UFPE)
Dorgival Guedes (UFMG)
Edmundo Madeira (UNICAMP)
Eduardo Alchieri (UnB)
Eduardo Cerqueira (Universidade Federal do Pará)
Elias Duarte Jr. (UFPR)
Elmano Ramalho Cavalcanti (Universidade Federal de Campina Grande)
Evaldo Silva (Universidade Salgado de Oliveira)
Ewerton Salvador (UFMG)
Fabíola Greve (Universidade Federal da Bahia)
Fabio Kon (IME – USP)
Fabio Vieira (Universidade Federal do Estado do Rio de Janeiro)
Fatima Duarte-Figueiredo (PUC Minas)
Fábio Luciano Verdi (Universidade Federal de São Carlos, Campus Sorocaba)
Fernanda Sumika Souza (UFSJ)
Fernando Dettoni (PPGCC-UFSC)
Fernando Dotti (PUCRS)
Fernando Pedone (University of Lugano, Switzerland)
Fernando Ramos (University of Lisbon)
Flavia Delicato (UFRJ)
Francisco Brasileiro (Universidade Federal de Campina Grande)
Francisco Silva (Universidade Federal do Maranhão)
Frederico Lopes (UFRN)
Geraldo Robson Mateus (UFMG)
Gilvan Durães (Instituto Federal Baiano)
Glederson Santos (Instituto Federal Sul-Rio-Grandense)
Gustavo Figueiredo (Universidade Federal da Bahia)
Gustavo Pavani (UFABC)
Horácio Oliveira (Universidade Federal do Amazonas)
Humberto Marques (PUC Minas)
Hylson Netto (Universidade Federal de Santa Catarina)
Igor Dos Santos (Universidade Federal do Rio de Janeiro)
Igor Moraes (Universidade Federal Fluminense)
Iguatemi E. Fonseca (Universidade Federal da Paraíba)

Islene Garcia (UNICAMP)
Ítalo Cunha (Universidade Federal de Minas Gerais)
Jacir Bordim (Universidade de Brasília)
Jair Leite (Universidade Federal do Rio Grande do Norte)
Jéferson Nobre (UFRGS)
Jó Ueyama (Universidade de São Paulo)
João Vitor Torres (Universidade Federal do Rio de Janeiro)
Joni da Silva Fraga (UFSC)
José De Souza (UFC)
José Ferreira de Rezende (UFRJ)
José Augusto Suruagy Monteiro (UFPE)
Jose-Marcos Nogueira (Universidade Federal de Minas Gerais)
Juliano Naves (Universidade Federal Fluminense)
Katia Obraczka (University of California Santa Cruz)
Kelvin Dias (UFPE)
Kleber Cardoso (Universidade Federal de Goiás)
Lau Cheuk Lung (UFSC)
Leandro Magnabosco (UFSC)
Leandro Resendo (IFES)
Leandro Villas (UNICAMP)
Leônidas Lima Junior (Federal University of Pernambuco)
Leobino Sampaio (Universidade Federal da Bahia)
Leonardo Oliveira (UFMG)
Leopoldo Mauricio (Universidade Federal do Rio de Janeiro)
Liane Margarida Rockenbach Tarouco (UFRGS)
Lisandro Zambenedetti Granville (UFRGS)
Luci Pirmez (UFRJ)
Luciana Rech (Universidade Federal de Santa Catarina)
Luciano Barreto (UFSC)
Luciano Paschoal Gaspary (UFRGS)
Lucio Rocha (Federal University of São Carlos)
Luis Rodrigues (INESC-ID/IST, Portugal)
Luis Carlos De Bona (Universidade Federal do Paraná)
Luis Henrique Costa (Universidade Federal do Rio de Janeiro)
Luiz Eduardo Buzato (UNICAMP)
Luiz Fernando Bittencourt (UNICAMP)
Luiz Filipe Vieira (UFMG)
Lyno Ferraz (UFRJ)
Magnos Martinello (UFES)
Marcel William Rocha da Silva (Universidade Federal Rural do Rio de Janeiro)
Marcelo Dias de Amorim (LIP6/CNRS - UPMC Sorbonne Universités, France)
Marcelo Pitanga (Universidade Federal do Rio de Janeiro)
Marcelo Rubinstein (Universidade do Estado do Rio de Janeiro)
Marcelo Santos (Universidade Federal de Pernambuco)
Marco Spohn (Universidade Federal da Fronteira Sul)
Marcos Pinheiro (UFRN)
Marcos Salvador (Lenovo)
Marcos Vieira (UFMG)
Marcus de Lima Braga (COPPE/UFRJ)
Marcus Sandri (Federal University of Sao Carlos)

Marinho Barcellos (UFRGS)
Martin Andreoni Lopez (Universidade Federal do Rio de Janeiro)
Mateus Augusto Silva Santos (Universidade Estadual de Campinas)
Mauro Fonseca (UTFPR)
Michael Stanton (RNP and UFF)
Michele Nogueira (Universidade Federal do Paraná)
Michelle Wangham (Universidade do Vale do Itajaí)
Miguel Elias Mitre Campista (Universidade Federal do Rio de Janeiro)
Moises Ribeiro (Universidade Federal do Espírito Santo)
Nabor Mendonca (UNIFOR)
Natalia Castro Fernandes (Universidade Federal Fluminense)
Nazareno Andrade (Universidade Federal de Campina Grande)
Noemi Rodriguez (PUC-Rio)
Odorico Mendizabal (Universidade Federal do Rio Grande – FURG)
Otto Carlos Muniz Bandeira Duarte (UFRJ)
Paulo Aguiar (UFRJ)
Paulo Cunha (UFPE)
Paulo Mafra (UFSC)
Paulo Pires (UFRJ)
Paulo Rego (Federal University of Ceará)
Paulo Sausen (UNIJUÍ - Universidade Regional do Noroeste do Estado do RGS)
Paulo André da Silva Gonçalves (UFPE)
Pedro Velloso (Universidade Federal do Rio de Janeiro)
Pooyan Jamshidi (Dublin City University, Ireland)
Rafael Lopes Gomes (UNICAMP)
Rafael Obelheiro (UDESC)
Rafael Pasquini (UFU)
Raimundo Jose de Araujo Macedo (UFBA)
Raquel Mini (Pontifícia Universidade Católica de Minas Gerais)
Renato de Moraes (UFPE)
Renato Elias Nunes Moraes (Universidade Federal do Espírito Santo)
Ricardo Pfitscher (Universidade do Estado de Santa Catarina)
Ricardo Salgueiro (Universidade Federal de Sergipe)
Ricardo Luis dos Santos (Universidade Federal do Rio Grande do Sul)
Robespierre Pita (Universidade Federal da Bahia)
Rodolfo Villaça (Universidade Federal do Espírito Santo)
Rodrigo de Souza Couto (Universidade do Estado do Rio de Janeiro)
Rodrigo Fonseca (Brown University)
Rodrigo Tessinari (Universidade Federal do Espírito Santo)
Ronaldo Ferreira (Universidade Federal de Mato Grosso do Sul)
Ronaldo Salles (Instituto Militar de Engenharia)
Rossana Andrade (Universidade Federal do Ceará)
Sérgio Cardoso (Instituto Militar de Engenharia)
Sidney Lucena (Universidade Federal do Estado do Rio de Janeiro – UNIRIO)
Silvana Rossetto (UFRJ)
Stenio Fernandes (Universidade Federal de Pernambuco)
Taniro Rodrigues (UFRN)
Thais Vasconcelos Batista (UFRN)
Thiago Genez (UNICAMP)
Vinícius Guimarães (Instituto Federal Sul-Rio- Grandense - Campus Charqueadas)

Vinicius Cunha M Borges (Federal University of Goiás)
Vinicius Mota (Universidade Federal de Minas Gerais)
Vinod Rebello (Universidade Federal Fluminense)
Vitor Borges Coutinho da Silva (Universidade Federal do Rio de Janeiro)
Wagner Meira Jr. (UFMG)
Weverton Cordeiro (Federal Institute of Pará at Itaituba)
William Giozza (UnB)

Índice

Políticas para Seleção Assistida de Máquinas Virtuais em Nuvem de Computadores

Mario Meireles Teixeira[1], **Azer Bestavros**[2]

[1]Departamento de Informática – Universidade Federal do Maranhão
65080-805 – São Luís, MA – Brasil

[2]Computer Science Department – Boston University
Boston, MA 02215 – USA

mario@deinf.ufma.br, best@bu.edu

Abstract. *In this paper we propose an end-to-end approach to the VM allocation problem using policies based uniquely on round-trip time measurements between VMs. We propose and implement end-to-end algorithms for VM selection that cover desirable profiles of communications between VMs in distributed applications in a cloud setting. The use of informed VM selection allowed to select instances with an average RTT up to 85% lower, in one case, and to find local and global centers of VM clusters in another. The approach described is completely independent of cloud architecture and is adaptable to different types of applications and workloads, as well as light and transparent for cloud providers.*

Resumo. *Este artigo propõe uma abordagem fim a fim para o problema de alocação de VMs em nuvens de computadores, usando políticas baseadas exclusivamente em medições de tempo de ida e volta entre VMs. São propostos algoritmos fim a fim para seleção de VMs que consideram perfis típicos de comunicação entre VMs executando aplicações distribuídas em nuvem. O uso da seleção assistida de VMs permitiu selecionar instâncias com um RTT médio até 85% menor, em um caso, bem como localizar centros locais e globais de aglomerados de VMs, em outro. A abordagem descrita é completamente independente da arquitetura da nuvem e é adaptável a diferentes tipos de aplicações e cargas de trabalho, além de leve e transparente para os provedores de nuvem.*

1. Introdução

Em cenários atuais de utilização de nuvem, na modalidade IaaS, um cliente solicita e recebe do provedor de nuvem um conjunto de máquinas virtuais (VMs) com capacidade determinada, que devem ser organizadas de forma a proporcionar os melhores serviços possíveis com o mínimo consumo de recursos e, portanto, com o menor custo para o cliente.

Quando os clientes de nuvem solicitam um conjunto de VMs de um provedor de nuvem pública, como a Amazon EC2[1], eles geralmente não têm qualquer informação sobre em qual máquina física cada VM está de fato localizada e nem sobre como esses recursos físicos compartilhados são utilizados por outros clientes da nuvem, em termos

[1]http://aws.amazon.com/

de consumo de CPU, dados ou rede. Além disso, os clientes não sabem como as VMs que lhes foram atribuídas relacionam-se quanto ao atraso dentro da rede da nuvem.

Neste contexto, este trabalho propõe um *Serviço de Recomendação* para ser usado por clientes de nuvem que desejam fazer uma melhor utilização dos recursos por eles contratados, assistindo-lhes em suas decisões de alocação de VMs, de modo a não desperdiçar nem abusar desses recursos. Essas VMs seriam selecionadas a partir de um conjunto maior de VMs atribuído pelo provedor de nuvem ao cliente, ou a um agregador ou *broker* operando em nome de um conjunto de clientes [Bestavros and Krieger 2014].

O foco principal não está na alocação de VMs sob a perspectiva do administrador de nuvem, pois este problema já foi bastante estudado recentemente [Alicherry and Lakshman 2012] [Meng et al. 2010] [Piao and Yan 2010]. Em vez disso, nosso interesse particular é na capacitação dos clientes da nuvem, que têm utilizado seus recursos alocados como uma caixa preta, na esperança de que os provedores de nuvem saibam o que é 'melhor' para eles.

Outra importante motivação resulta do fato que nuvens não são mais apenas infraestruturas em que se executam aplicações de *Big Dat*a ou de computação de alto desempenho. Cada vez mais, a nuvem hospeda aplicações que exigem comunicação intensiva, tais como jogos, serviços multimídia e aplicações de tempo real, para as quais problemas de latência (e variação desta) são de suma importância. Problemas de latência de comunicação podem alongar os tempos de conclusão de tarefas na nuvem [Alicherry and Lakshman 2012], prejudicando a qualidade dos serviços prestados. Além disso, pesquisas recentes têm mostrado que os padrões de intercomunicação entre máquinas virtuais podem desempenhar um papel importante e decisivo na eficiência das aplicações [LaCurts et al. De 2013] e, definitivamente, ter um impacto sobre seu desempenho e capacidade de resposta percebidos pelos usuários finais.

Mais especificamente, este trabalho trata sobre algoritmos de alocação de VMs que empregam tempos de ida e volta (RTT) como heurística para auxiliar a alocação de VMs em nuvens IaaS. Medições em tempo real de atrasos entre máquinas virtuais são usadas como entrada para o Serviço de Recomendação, que tem a responsabilidade de selecionar um subconjunto m de VMs, dentre as n possíveis, sujeito a certos requisitos colocados pelo cliente da nuvem.

A abordagem empregada é fim a fim e independente da topologia interna da nuvem. Esta é uma inovação e também um diferencial em relação a pesquisas semelhantes [Meng et al. 2010] [Alicherry and Lakshman 2012], que geralmente empregam algum conhecimento da arquitetura subjacente do *datacenter* com este propósito. Argumenta-se que esta abordagem é consistente com modelos de negócio emergentes em nuvem e inter-nuvens, como no caso da iniciativa *Massachusetts Open Cloud* [MOC 2014].

Este artigo está assim organizado: a Seção 2 discute o cenário atual de utilização de nuvens e descreve o Serviço de Recomendação. A Seção 3 apresenta as políticas de seleção de VMs e os algoritmos propostos. Na seção 4, os algoritmos e seus resultados são avaliados através de simulação, utilizando-se uma arquitetura de referência validada a partir de medições reais na nuvem Amazon EC2. Por fim, a Seção 5 discute as principais conclusões e trabalhos futuros.

2. Arquitetura do Sistema

2.1. Cenário

Clientes de serviços de infraestrutura de computação em nuvem (IaaS) normalmente acessam um provedor de nuvem pública através de uma interface web e requisitam máquinas virtuais genéricas, que pretendem empregar em diferentes tipos de tarefas computacionais. A ideia é que os recursos disponíveis sejam encarados de forma utilitária (como se fossem energia, água, etc.), pois os clientes não têm interesse em um aplicativo em nuvem específico, mas apenas em ter acesso a uma plataforma de computação remota para finalidades as mais diversas.

Muitos provedores de nuvem disponibilizam algum tipo de serviço de monitoramento para seus clientes, através do qual estes podem obter informações de alto nível e/ou refinadas sobre como suas aplicações estão usando as VMs que lhes foram atribuídas, em termos de consumo de CPU, dados e rede. No entanto, estes serviços de monitoramento não fornecem qualquer visibilidade sobre o impacto de aplicativos de terceiros executando paralelamente em outras VMs, seja na mesma máquina física ou em outro lugar da nuvem. A falta desta informação pode conduzir a mapeamentos de aplicações para VMs completamente inadequados. Por exemplo, caso se deseje selecionar uma VM para ser um servidor, é interessante que esta se localize em uma posição equidistante dos seus clientes dentro da nuvem ou, ainda, pode-se querer que o tempo de obtenção da informação seja inferior a um certo limiar. Numa seleção aleatória de VMs, fica difícil senão impossível atender a estes requisitos, o que é perfeitamente viável com a aplicação da seleção assistida proposta.

O objetivo é fornecer um serviço que irá coletar informações de uso da nuvem, de um lado e associá-las com as necessidades do cliente, de outro, a fim de sugerir um mapeamento de VMs em conformidade com os requisitos colocados e adequado à utilização atual dos recursos da nuvem. Esse serviço seria especialmente útil em modelos de negócios emergentes na computação em nuvem, como mostrado na Figura 1, onde se veem corretores (*brokers*) que contratam um lote de recursos do provedor de nuvem e atuam como revendedores (ou intermediários) entre o provedor de nuvem e o cliente de nuvem [Bestavros and Krieger 2014] [Böhm et al. 2010].

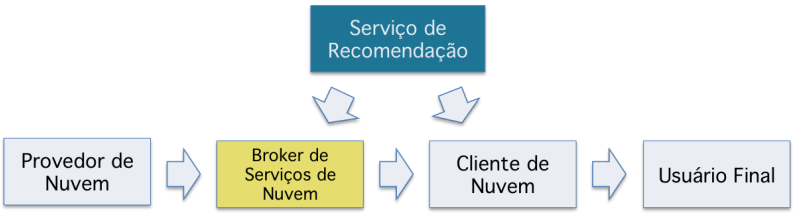


Figura 1. Modelos de negócio emergentes em Computação em Nuvem

2.2. Serviço de Recomendação

Este trabalho propõe um **Serviço de Recomendação**, como mostrado na Figura 2, ao qual os clientes da nuvem fazem uma solicitação de VMs que deve atender a requisitos específicos (1), como um limite mínimo de tempo de ida e volta (*Round-trip time* – RTT) entre as VMs ou uma topologia particular de VMs dentro da nuvem.

O Serviço de Recomendação mantém um conjunto de n VMs anteriormente contratadas a partir de um provedor de nuvem e periodicamente monitora seu desempenho (2). Quando um pedido chega, este serviço irá selecionar um subconjunto de m VMs das n totais que deve satisfazer as demandas do cliente. Em seguida, sugere este arranjo de VMs ao solicitante (3), que por sua vez distribui suas aplicações nas VMs. Consequentemente, o cliente pode escolher entre os recursos disponíveis com mais certeza, realizando a atribuição de suas aplicações às VMs de forma mais esclarecida.

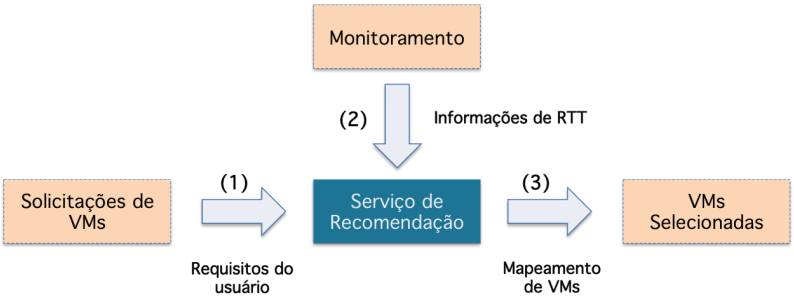


Figura 2. Arquitetura do Serviço de Recomendação

O Recomendador não funciona como um *broker*, uma vez que ele não realiza alocação de VMs. O que ele realmente faz é monitorar o ambiente ao redor de cada VM (por exemplo, a latência de rede ou *throughput*), a fim de ser capaz de melhor balancear as solicitações de seus clientes frente aos recursos disponíveis, proporcionando uma seleção de VMs adequada às suas necessidades.

É nossa percepção que este tipo de serviço pode se tornar uma peça central na gestão de recursos de nuvem em um futuro próximo, na medida em que os aplicativos tornam-se mais e mais complexos e com exigências mais rigorosas, que não podem ser satisfeitas unicamente pelos modelos de negócio atuais do tipo provedor-a-cliente.

Uma parte fundamental do Serviço de Recomendação são os algoritmos de seleção de VMs e também as informações de desempenho disponíveis para eles. O presente estudo detém-se particularmente em algoritmos que procuram tirar proveito das informações de latência fim a fim entre VMs para suas decisões de alocação.

3. Algoritmos de Seleção de VMs

Esta seção detalha a abordagem para monitoramento de latência fim a fim entre VMs e propõe 4 algoritmos para Seleção Assistida de máquinas virtuais, os quais usam algum tipo de informação de atraso para as suas sugestões de alocação.

3.1. Matriz de Latências

A entrada principal para os algoritmos consiste em uma *Matriz de Latências* ($RTT_{i,j}$), que é atualizada periodicamente com informações de tempo de ida e volta (RTT) entre qualquer par (i, j) de VMs conhecido (ou gerenciado) pelo Serviço de Recomendação:

$$RTT_{i,j} = \begin{bmatrix} 0 & t_{1,2} & t_{1,3} & \dots & t_{1,n} \\ t_{2,1} & 0 & t_{2,3} & \dots & t_{2,n} \\ t_{3,1} & t_{3,2} & 0 & \dots & t_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ t_{n,1} & t_{n,2} & t_{n,3} & \dots & 0 \end{bmatrix} \quad (1)$$

Cada linha ou coluna na matriz representa uma VM dentro da nuvem e o elemento (i, j) contém o tempo de ida e volta entre a VM$_i$ e a VM$_j$. Como esta informação é simétrica, o elemento (j, i) equivalente é ignorado e os valores apresentados na coluna principal da matriz são ajustados em zero. Para fins de implementação, assume-se uma matriz triangular superior.

Optou-se por usar a latência como base para os algoritmos de seleção de VMs porque o RTT é uma métrica confiável para medir a distância de rede dentro de nuvens, como apontado por trabalhos recentes [Alicherry and Lakshman 2012] [Battré et al. 2011] [Jonglez et al. 2014] [LaCurts et al. 2013] [Shea et al. 2014]. Além disso, RTTs são computacionalmente baratos e simples de usar e obter. Neste estudo, medições de latência serão utilizadas para estimar a distância entre VMs e constituir-se-ão em um parâmetro fundamental para as recomendações de alocação de VMs.

Para verificar como isso funcionaria num cenário real, foi realizado um experimento de monitoramento de tempos de ida e volta utilizando-se instâncias Amazon EC2. Primeiramente, colocou-se um aplicativo do tipo PING, cliente-servidor, em cada uma das VMs contratadas. O cliente PING acorda periodicamente e envia uma mensagem a cada uma das outras máquinas virtuais, em *round robin*. Este passo é repetido várias vezes (definido como 10 nos experimentos); quando o cliente recebe as mensagens de retorno, calcula-se o atraso médio para cada par (VM$_i$, VM$_j$) e atualizam-se os valores na Matriz de Latências.

3.2. Algoritmo Colmeia (*Hive*)

Este primeiro algoritmo tem como objetivo encontrar um conjunto de VMs próximas umas das outras por um tempo de ida e volta menor do que r ms, ou seja, o RTT entre qualquer par de VMs pertencente a esse conjunto deve ser inferior a r. Uma variação disto seria exigir que o percentil 0.95 das medições dos RTT_{ij} fosse inferior ao limite r, a fim de evitar que um arranjo adequado de VMs fosse descartado por causa de alguns valores discrepantes.

Esta abordagem é denominada de *Algoritmo Colmeia*, uma vez que tem a intenção de encontrar aglomerados de VMs próximas em uma tentativa de explorar localidade de referência. Na verdade, a maioria dos aplicativos na nuvem pode se beneficiar de tal abordagem, pois ela minimiza o tráfego intra-nuvem e como resultado reduz o uso de banda, além de ajudar a diminuir a fragmentação do *datacenter*, uma vez que as VMs cooperando em uma tarefa tendem a ficar próximas entre si. No entanto, observe que latência, neste caso, está em geral mas não necessariamente relacionada com distância física na rede. Apesar de VMs no mesmo processador ou *rack* tenderem a ter valores de RTT menores entre elas, em alguns casos isso pode não ser verdade devido a questões de virtualização ou de carga de trabalho. Por conseguinte, pode ser que uma VM em outro *rack* venha a ser o vizinho mais próximo.

O Algoritmo Colmeia é muito difícil de se implementar em termos práticos. Ele é de fato um caso especial de um problema NP-difícil em grafos. O que se busca encontrar é um subgrafo onde só é possível desenhar uma aresta de qualquer nó i para qualquer nó j se o RTT entre eles for menor do que o limite especificado. Quanto maior o número m de VMs na colmeia, maior a complexidade do algoritmo. Se a intenção for encontrar a maior colmeia possível dado que $RTT_{ij} < distance$ então ter-se-ia o problema MAXCLIQUE, onde o desafio é encontrar um clique de tamanho máximo, o que é provado NP-difícil.

Uma maneira de resolver este problema é restringir o número de VMs na colmeia para um valor muito pequeno. Na verdade, na Seção 3.5 é apresentado um exemplo de uma abordagem simples para encontrar m pares de VMs dentro da nuvem com um RTT entre elas menor do que um certo limite.

3.3. Algoritmo Estrela (*Star*)

Nossa segunda abordagem, o *Algoritmo Estrela* procura encontrar a resposta para uma pergunta simples: dado um conjunto de VMs dentro da nuvem, deseja-se saber qual VM encontra-se aproximadamente no centro do conjunto, considerando-se a latência. Em outras palavras, o objetivo é organizar este conjunto de VMs de acordo com uma topologia em estrela, cujo centro (a VM selecionada) está mais perto, em média, de todos os seus vizinhos.

O Algoritmo 1 a seguir retorna, para um conjunto de VMs, aquela que possui a menor média de RTTs em relação a todas as outras VMs do conjunto, isto é, o 'centro' da estrela. O algoritmo percorre a diagonal principal da matriz, onde cada iteração se refere a uma VM específica. A matriz considerada é do tipo triangular superior, pois assim aproveita-se a simetria dos tempos de ida e volta (linhas 2 e 3) e a matriz pode, então, ocupar apenas metade da memória requerida por $n \times n$ elementos. A complexidade deste algoritmo é claramente $O(n^2)$.

Algorithm 1 Star

Input: RTT : Matriz de Latências para n VMs

1: **for** $i = 0$ **to** <*num. linhas*> *de* RTT **do**
2: $sumx \leftarrow \sum_{j=i+1}^{n} t_{ij}$
3: $sumy \leftarrow \sum_{j=0}^{i-1} t_{ji}$
4: $avg \leftarrow (sumx + sumy)/(n-1)$
5: **if** avg é a menor média até agora **then**
6: $center \leftarrow i$
7: **end if**
8: **end for**
9: **return** $center$

Este algoritmo pode ser útil para qualquer aplicação distribuída em que um nó deve estar no ponto médio em relação aos outros nós do conjunto. Este poderia ser o caso de uma VM funcionando como um diretório de nomes, um serviço de distribuição de chaves ou transmitido um fluxo de dados multicast para outras VMs. Em um modelo convencional provedor-a-cliente, onde ao último é simplesmente atribuída uma certa quantidade de VMs, é geralmente muito difícil acertar qual das VMs deveria ser escolhida como ponto central (*hub*). Vê-se, portanto, que em todos esses casos o cliente da

nuvem poderia beneficiar-se claramente desta **Seleção Assistida** de VMs, baseada em informações de latência periodicamente coletadas.

3.4. Algoritmo Melhores Centros (*Best Centers)*

Generalizando a abordagem do Algoritmo Estrela, pode-se desejar escolher não apenas uma VM como o centro, mas sim encontrar as m melhores VMs que poderiam funcionar como hubs (ou super-nós), agregando o tráfego de VMs adjacentes dentro da nuvem. Como exemplo de situações em que isso é necessário, considere uma aplicação que exige a manutenção de uma rede *overlay* ou uma tabela hash distribuída com super-nós utilizados para o gerenciamento de clusters próximos.

Propõe-se o Algoritmo 2 que é capaz de encontrar m centros dentre n VMs, todos os quais têm o menor RTT médio em relação a todas as outras VMs no conjunto. O algoritmo *Melhores Centros* tem como objetivo identificar as VMs que são as melhores candidatas a estrela e, portanto, podem potencialmente se tornar centros em torno dos quais grupos de VMs seriam formados. Assim, é possível identificar 'colmeias' de VMs espalhadas por toda a nuvem, a fim de responder a necessidades específicas dos clientes.

Algorithm 2 BestCenters

Input: RTT : Matriz de Latências para n VMs; m : número de 'best centers'

1: **for** $i = 0$ **to** <*num. linhas*> *de* RTT **do**
2: $\quad sumx \leftarrow \sum_{j=i+1}^{n} t_{ij}$
3: $\quad sumy \leftarrow \sum_{j=0}^{i-1} t_{ji}$
4: $\quad avg \leftarrow (sumx + sumy)/(n-1)$
5: $\quad centers\{\} \leftarrow [i, avg]$
6: **end for**
7: $best\{\} \leftarrow centers\{\}$ sorted by key avg ascending
8: **return** $best\{0..(m-1)\}$ {Retorna os m melhores centros}

Observe que este algoritmo pode ser usado como uma abordagem heurística para resolver o problema de agrupamento introduzido na Seção 3.2, porém utilizando uma técnica com muito menos complexidade computacional. Como pode ser visto, o Algoritmo 2 tem uma complexidade que é, pelo menos, igual à do algoritmo Estrela, ou seja, $O(n^2)$, além de passos adicionais para classificar o array $centers$, o que para uma rotina do tipo Quicksort seria $O(n \log n)$, em média.

3.5. Algoritmo Vizinhos *(Neighbors)*

Considere agora o caso em que se pretende encontrar m pares de VMs dentro da nuvem muito próximos uns dos outros, tipicamente com um tempo de ida e volta entre eles abaixo de um determinado limiar. Este é um caso especial do algoritmo Colmeia, porém limitando o número de VMs na colmeia para apenas duas, a fim de evitar uma explosão no número de iterações.

Esta abordagem é descrita no Algoritmo 3, que recebe uma matriz de latências como entrada e devolve todos os pares de VMs cujos RTTs sejam inferiores a um determinado limiar. Como a informação de tempo de ida e volta é simétrica, só é preciso verificar os elementos do lado direito da diagonal principal ($i < j$, como na Linha 2), uma

vez que os à sua esquerda serão testados à medida que o loop avança ao longo da matriz. Isto ainda proporciona uma complexidade $O(n^2)$ para o algoritmo, no entanto permite-lhe trabalhar com apenas metade dos elementos.

Algorithm 3 Neighbors

Input: RTT : Matriz de Latências para n VMs; $thres$: RTT máximo entre VMs

```
1: for i = 0 to <num. linhas> de RTT() do
2:    for j = i + 1 to <num. colunas> de RTT() do {somente i < j}
3:       if RTT(i, j) < thres then
4:          pairs{} ← RTT(i, j)
5:       end if
6:    end for
7: end for
8: return pairs{}
```

4. Avaliação das Políticas Fim a Fim

4.1. Arquitetura de Referência

A fim de avaliar a eficácia das políticas de seleção assistida de VMs e dos algoritmos propostos, foram realizados extensos estudos de simulação usando um modelo para latências entre VMs, desenvolvido no escopo deste trabalho a partir de medições reais entre instâncias EC2 da nuvem *Amazon Web Services* (AWS).

O primeiro passo foi definir uma arquitetura de referência da nuvem a ser utilizada para validar os algoritmos. Em *datacenters* atuais, a arquitetura de rede interna da nuvem é normalmente organizada de forma hierárquica, como mostrado na Figura 3. Na camada inferior, veem-se os *racks*, cada um com vários nós computacionais (referidos como CPUs, por simplicidade) que servem como hosts para um certo número de máquinas virtuais. As VMs dentro da mesma CPU comunicam-se diretamente, sem qualquer interferência da rede. Já as máquinas virtuais localizadas no mesmo rack, mas em diferentes processadores precisam se comunicar através de um switch *Top-of-Rack* (ToR). As VMs em racks diferentes têm que utilizar um agregador para alcançar uma a outra, por exemplo, se uma VM no rack 1 deseja enviar uma mensagem para outra VM no rack 2, esta mensagem terá de passar pelo switch ToR do primeiro rack, em seguida pelo switch agregador de segundo nível, o switch ToR do rack de destino, e assim por diante. Se os racks pertencerem a diferentes agregados, então um agregador de primeiro nível será também envolvido na comunicação. Assim, quanto mais longe estiverem as VMs fisicamente, maior tende a ser a latência de comunicação entre elas. Apesar de outros fatores como a sobrecarga da CPU hospedeira poderem influenciar na latência de rede, aplicações de nuvem em geral podem beneficiar-se de políticas de seleção de VMs que visem assegurar que a comunicação ocorra entre VMs próximas umas das outras.

Para efeito de validação dos algoritmos de seleção de VM, cientes de localização (*locality-aware*), foi utilizada a arquitetura de nuvem descrita acima. Nos experimentos, esta foi configurada com um agregador de nível superior ao qual 2 outros agregadores foram conectados, cada um com 2 racks ligados a ele, com 4 CPUs e 10 máquinas virtuais em cada uma. Isto dá um total de 160 VMs na nuvem (10 VMs por CPU, 4 CPUs por rack, 4 racks).

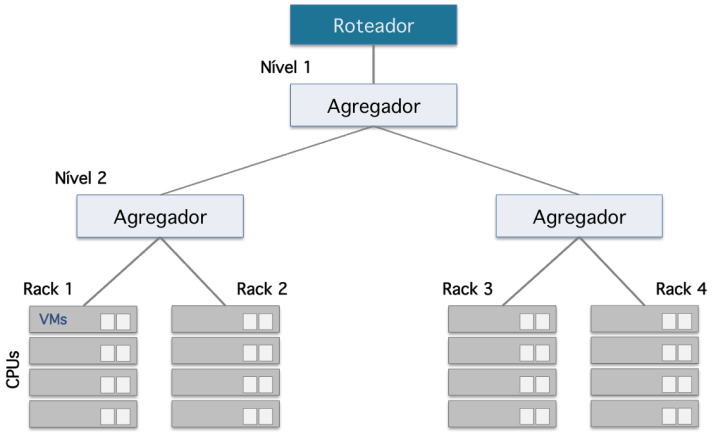


Figura 3. Arquitetura de nuvem usada como referência

Tabela 1. Diferentes níveis para os tempos de ida e volta

Nível	Localização da VM	Peso	%
1	Mesma CPU	0,1	5,6%
2	Mesmo Rack	1	18,8%
3	Mesmo Agregado	10	25,1%
4	Agregados distintos	50	50,3%

4.2. Amostragem dos Tempos de Ida e Volta (RTT)

Como discutido na Seção 3.1, os elementos da *Matriz de Latências* informam a distância, ou seja, os tempos de ida e volta entre quaisquer duas VMs dentro da nuvem. Para fins de implementação, é irrelevante se os dados na matriz referem-se a toda a nuvem ou a apenas um subconjunto de VMs alocadas a um cliente pelo provedor de nuvem. No contexto do Serviço de Recomendação (Seção 2.2), é mais realista assumir que se está lidando com um subconjunto de VMs sob controle de um *broker* de serviços de nuvem (Figura 1), que irá utilizar os algoritmos propostos para fazer uma seleção informada de VMs e então repassá-las ao cliente interessado.

A fim de preencher a Matriz de Latências, foi empregada uma distribuição uniforme no intervalo $[0, 1)$ com o objetivo de gerar aleatoriamente os valores de tempo de ida e volta. Estes serviram, então, como entrada para uma função de distribuição de probabilidade (FDP) acumulada a fim de produzir valores de RTT mais condizentes com os de uma nuvem típica. Os valores resultam, portanto, estratificados em níveis distintos, conforme as VMs comunicantes estejam na mesma CPU, rack, agregado, e assim por diante.

A Tabela 1 enumera os quatro níveis diferentes de RTT considerados, dependendo de onde as VMs estão localizadas. Os pesos atribuídos a cada nível foram consistentes com os experimentos realizados em máquinas Amazon EC2, como relatado na Seção 3.1. A última coluna indica a percentagem de VMs presentes em cada categoria, derivada da arquitetura mostrada na Figura 3. Esta tabela é empregada para preencher a Matriz de Latências, usada nas simulações.

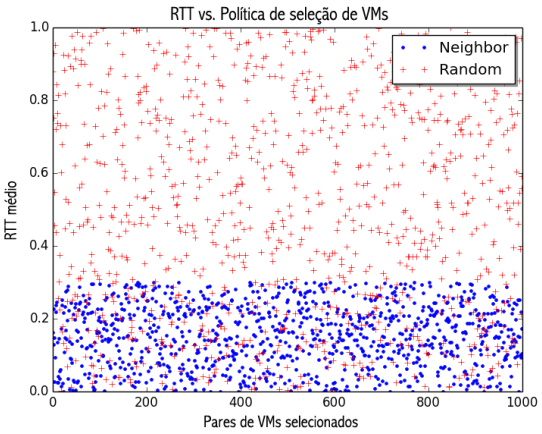


Figura 4. Comparação da seleção de VMs: algoritmo Vizinhos vs. Aleatório

4.3. Resultados

Nesta seção, serão relatados alguns resultados obtidos com o emprego das políticas propostas com o intuito de realizar uma seleção de m VMs a partir de uma população de n máquinas. A principal entrada para os algoritmos de seleção é a Matriz de Latências conforme descrito nas seções 3.1 e 4.2. Cada algoritmo é comparado com o caso aleatório, ou seja, quando um cliente simplesmente atribui aplicações a VMs ao acaso, como acontece normalmente em cenários atuais de utilização de nuvem. As simulações foram realizadas em um MacBook com processador Intel i5, 4GB de memória e sistema operacional Mac OSX 10.10.

No primeiro experimento, foi comparado o desempenho do **Algoritmo Vizinhos** com uma seleção aleatória de VMs. A Matriz de Latências é preenchida usando-se uma distribuição uniforme no intervalo $[0, 1)$ e o limiar de RTT é fixado em 0,3 unidades de tempo. O algoritmo foi executado para 100 VMs (ou seja, uma matriz de valores de RTT 100×100) com o objetivo de encontrar 1.000 pares de VMs dentro do limiar determinado. Os resultados na Figura 4 mostram que os valores de RTT para todos os pares de VMs selecionados pelo Algoritmo Vizinhos caem rigorosamente dentro da faixa de 0,0 a 0,3, como esperado, com um tempo médio de 0,1553. Para o caso aleatório, os pares selecionados encontram-se espalhados por todo o espaço de solução, uma vez que qualquer um deles poderia ser selecionado, exibindo um RTT médio de 0,5073. A percentagem de pares rejeitados pela abordagem Vizinhos, a fim de alcançar a seleção de VMs desejada é de 70,88%, em média, o que é consistente com uma distribuição uniforme entre 0 e 1 com um ponto de corte fixado em 0,3.

Para uma visão mais realista deste algoritmo em ação, foi então gerada uma matriz de latências de acordo com a FDP acumulada detalhada na Tabela 1. O limiar de RTT é fixado em 40, de modo que apenas os pares de VMs localizados no mesmo agregado, rack ou CPU sejam selecionados pelo algoritmo. O objetivo é novamente encontrar 1.000 pares de máquinas cujo RTT satisfaça os critérios especificados. Neste experimento, como pode ser visto através do extrato da Figura 5, os RTTs estão agrupados em torno dos limiares especificados pela FDP, no caso {0,1; 2; 30; 200 } ($nivel \times peso$). O RTT médio para os pares de VMs selecionados pelo algoritmo Vizinhos é 16,9566, em comparação com uma média de 112,6947 quando uma seleção aleatória é usada. A percentagem de descartes

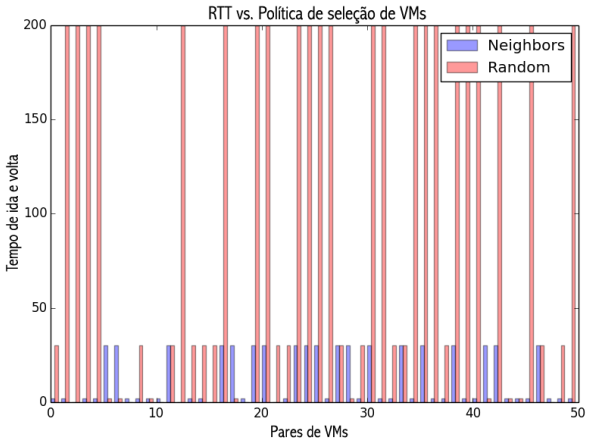


Figura 5. Comparação da seleção de VMs: Vizinhos vs. Aleatório (mesmo agregado)

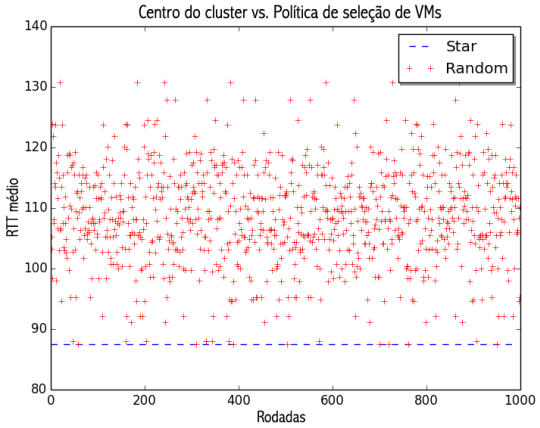


Figura 6. Comparação da seleção de VMs: Estrela vs. Aleatório

pela abordagem Vizinhos é de 49,44%, em média, o que corresponde a cerca de 50% de VMs localizadas em agregados distintos, conforme definido na Tabela 1. Para todos os casos relatados, o tempo de execução do algoritmo foi inferior a 1 segundo.

A segunda política de seleção a ser analisada é o **Algoritmo Estrela**. Neste caso, os valores para a matriz de latências foram gerados de acordo com a FDP descrita anteriormente. Consideram-se 100 VMs e executa-se o algoritmo para encontrar a VM que tem o menor RTT médio em relação a todos as outras VMs. Na Figura 6, há uma comparação entre o centro real encontrado pelo algoritmo Estrela e 1.000 escolhas aleatórias de centros. Como pode ser visto, o algoritmo proposto aponta sempre para o mesmo centro para uma dada configuração da matriz de RTTs, obtendo-se neste caso um RTT médio de 87,4828. No entanto, no caso aleatório, a maioria dos "centros" cai longe da meta, escapando-a por mais de 20%.

Finalmente, foi realizado um experimento para estudar o Algoritmo **Melhores Centros**. Foi gerada uma matriz de latências com 100 VMs usando uma FDP, como anteriormente. Este algoritmo atende a uma classe especial de aplicações que requerem um conjunto de *hubs* que de alguma forma condensam o tráfego ao seu redor, funcionando

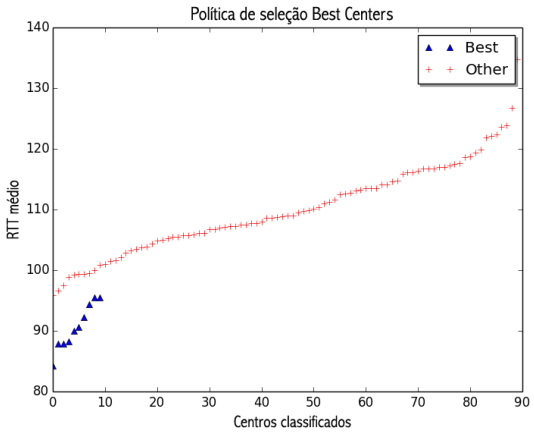


Figura 7. Seleção de VMs: algoritmo Melhores Centros

como pontos de atração. Neste experimento, pretende-se encontrar os 10% melhores centros pertencentes ao conjunto de VMs fornecido. Cada VM é escolhida, individualmente, como um centro candidato e seu RTT médio em relação às outras VMs é calculado. Finalmente, todos os RTT médios são classificados do menor para o maior.

Os resultados estão representados na Figura 7, onde os triângulos azuis indicam com precisão quais das VMs melhor servem ao propósito de funcionar como hubs locais. Ao contrário do que pode parecer à primeira vista, esses melhores centros não estão agrupados em uma região específica da matriz, mas sim espalhados por ela; por exemplo, em uma rodada particular do algoritmo a lista encontrada de candidatos a centro foi {36, 57, 25, 5, 82, 77, 50, 45, 75, 56}, sendo estes os índices das VMs na matriz. O tempo de execução do algoritmo foi inferior a um segundo, o que faz com que seja suficientemente rápido para ser utilizado em um cenário real de nuvem.

4.4. Discussão

Como mostrado pelos resultados acima, o emprego de políticas de seleção assistida de VMs permite harmonizar melhor o perfil dos aplicativos com a utilização atual das VMs e da nuvem. Isso resulta em um melhor mapeamento de aplicativos sobre VMs e pode ter um impacto decisivo no desempenho das aplicações executando na nuvem. Todos os algoritmos propostos neste trabalho atingiram o seu objetivo, com tempos de execução não significativos. Além disso, foram capazes de realizar escolhas de VMs muito mais adequadas do que aquelas obtidas por meio de uma seleção cega (aleatória) de recursos, como normalmente é feito.

Note-se que mesmo uma política de seleção simples e direta como o algoritmo Vizinhos pode realmente produzir resultados muito bons em termos de extrair um subconjunto de VMs que atenda a alguns requisitos anteriores especificados pelo cliente da nuvem. Este algoritmo pode ainda ser calibrado de modo a permitir apenas o retorno de VMs com uma determinada latência uma da outra, como foi o caso do experimento de 'mesmo agregado' relatado acima. No entanto, apesar do atraso estar fortemente correlacionado com a distância física dentro da nuvem, este nem sempre é o caso, porque às vezes uma latência mais alta resulta da sobrecarga da CPU causada pelo processamento da parte de rede da VM. De qualquer maneira, o que é requerido do ponto de vista do cli-

ente são pares de VMs com um atraso dentro de um determinado intervalo de tempo e é precisamente isso que o algoritmo fornece. Além disso, seu tempo de execução diminuto permite rodadas frequentes, a fim de responder rapidamente a mudanças nos padrões de comunicação entre aplicações dentro da nuvem.

Da mesma forma, os resultados do algoritmo Estrela são um excelente exemplo de como uma seleção assistida de VMs pode fazer uma grande diferença para os clientes da nuvem. Suponha que, por algum motivo, eles precisem escolher uma de suas VMs contratadas como centro de um dado subconjunto de VMs, talvez porque essa máquina deva executar um serviço de diretório. Muito provavelmente, o cliente iria terminar, por infortúnio, alocando o servidor de diretório longe de seus clientes, com um impacto negativo no desempenho do sistema. Por outro lado, este algoritmo pode indicar com precisão qual a VM mais apropriada para tal função. Melhor ainda, ele atinge essa conclusão por meio de medidas de RTT apenas, sem nenhum conhecimento adicional da arquitetura interna da nuvem, ou de quaisquer aplicações de terceiros executando simultaneamente nessas VMs.

O algoritmo Melhores Centros realiza uma tarefa ainda mais difícil, a de encontrar um conjunto de VMs, dentre as contratadas, que sejam mais adequadas para desempenhar o papel de centros de tráfego locais. Tal característica seria especialmente importante numa aplicação *peer-to-peer* dentro da nuvem, onde cada nó poderia assumir o papel de um supernó para as suas máquinas vizinhas. Como os resultados demonstraram, este algoritmo é capaz de encontrar as VMs mais adequadas para este fim e repassá-las ao cliente da nuvem, o que representa uma enorme vantagem quando comparado com uma seleção cega de VMs.

As políticas de seleção e algoritmos analisados distinguem-se de abordagens semelhantes porque são capazes de realizar melhores escolhas de VMs com um conhecimento mínimo da infraestrutura de nuvem subjacente. Eles simplesmente inferem as distâncias entre VMs a partir de medições de tempo de ida e volta e então utilizam esta informação como heurística para fazer seleções mais inteligentes de VMs.

5. Conclusão

Pesquisas recentes apontaram que os padrões de comunicação podem desempenhar um papel importante no desempenho geral dos aplicativos na nuvem, o que tem atraído muita atenção da comunidade para esse tema. Neste trabalho, propõe-se um Serviço de Recomendação ao qual um cliente da nuvem faz uma solicitação de VMs, sujeita a critérios específicos e recebe, como resposta, um conjunto de máquinas que satisfaça essas exigências. Este serviço permite que sejam tomadas decisões de alocação de VMs mais bem informadas e adequadas, em contraste com a alocação cega de recursos frequentemente utilizada na computação em nuvem.

Em particular, este estudo teve seu foco em políticas e algoritmos de seleção de VMs, cientes de localização, que utilizam a latência de rede entre VMs como heurística para apoiar decisões de alocação. Medições de tempo de ida e volta foram usadas como a principal entrada para os algoritmos e quatro diferentes políticas de seleção de VMs foram propostas e avaliadas. Todas elas empregam uma abordagem fim a fim e são independentes de detalhes de topologia da nuvem. O tempo de execução dos algoritmos foi não significativo e todos retornaram recomendações de VMs muito melhores do que as

obtidas a partir de uma seleção não-informada de máquinas.

Como trabalho futuro, pretende-se implantar o Serviço de Recomendação e os algoritmos desenvolvidos em uma infraestrutura de nuvem real, assim como avaliar a utilidade desta abordagem para aplicações sensíveis à latência. Isto irá permitir uma melhor compreensão do desempenho dos algoritmos numa situação real, bem como indicará novas formas de aplicá-los.

Agradecimentos

O primeiro autor gostaria de agradecer à CAPES pelo suporte financeiro concedido a este trabalho por intermédio do programa Ciência Sem Fronteiras, bem como ao Hariri Institute da Boston University por tê-lo acolhido como Professor Visitante no ano de 2014. Esta pesquisa foi ainda apoiada parcialmente por várias concessões do NSF (*National Science Foundation*), nos EUA, dentre elas PFI:BIC award #1430145, SaTC awards #1414119 e #1012798, CNS awards #1347522 e #1239021.

Referências

Alicherry, M. and Lakshman, T. (2012). Network aware resource allocation in distributed clouds. In *INFOCOM, 2012 Proceedings IEEE*, pages 963–971. IEEE.

Battré, D., Frejnik, N., Goel, S., Kao, O., and Warneke, D. (2011). Evaluation of network topology inference in opaque compute clouds through end-to-end measurements. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 17–24. IEEE.

Bestavros, A. and Krieger, O. (2014). Toward an open cloud marketplace: Vision and first steps. *Internet Computing, IEEE*, 18(1):72–77.

Böhm, M., Koleva, G., Leimeister, S., Riedl, C., and Krcmar, H. (2010). Towards a generic value network for cloud computing. In *Economics of Grids, Clouds, Systems, and Services (GECON)*, pages 129–140. Springer.

Jonglez, B., Boutier, M., and Chroboczek, J. (2014). A delay-based routing metric. *arXiv:1403.3488*.

LaCurts, K., Deng, S., Goyal, A., and Balakrishnan, H. (2013). Choreo: network-aware task placement for cloud applications. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 191–204. ACM.

Meng, X., Pappas, V., and Zhang, L. (2010). Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

MOC (2014). Massachusetts Open Cloud (MOC) . `http://www.bu.edu/moc`. [Online; accessed 3-Nov-2014].

Piao, J. T. and Yan, J. (2010). A network-aware virtual machine placement and migration approach in cloud computing. In *9th International Conference on Grid and Cooperative Computing (GCC), 2010*, pages 87–92. IEEE.

Shea, R., Wang, F., Wang, H., and Liu, J. (2014). A deep investigation into network performance in virtual machine based cloud environments. In *INFOCOM, 2014 Proceedings IEEE*, pages 1285–93. IEEE.

Estimando a Centralidade de Intermediação em Redes Ópticas de Telecomunicações

Silvana Trindade[1], **Pedro Augusto Pereira Borges**[1,2], **Claunir Pavan**[1]

[1] Curso de Graduação em Ciência da Computação

[2]Curso de Graduação em Matemática

Universidade Federal da Fronteira Sul (UFFS) – Chapecó, SC-Brasil

syletri@gmail.com,{pedro.borges,claunir.pavan}@uffs.edu.br

Abstract. *From a set of real optical telecommunication network topologies, an expression is proposed to estimate the betweenness centrality of a network without prior knowledge of its topological structure, having only the number of nodes and links as requirement. Betweenness centrality shows the number of times each node is present in the shortest paths between other nodes. This information is relevant when planning networks, as the topology can have a significant impact on its costs. In this context, the analytical models can be used in evaluation tools as elimination criterion and to forecast costs, accelerating decision making. Experimental results reveal that the proposed expression obtain the value of the betweenness centrality with an average error less than* 3%.

Resumo. *A partir de um conjunto de topologias de redes ópticas reais de telecomunicações, é proposta uma expressão para a estimação da centralidade de intermediação de uma rede, sem a necessidade de conhecer a sua estrutura topológica, apenas com a informação do número de nós e enlaces. Esta medida indica a quantidade de vezes que os nós estão presentes ao longo dos caminhos mínimos entre outros nós. Esta informação é relevante no planejamento preliminar de redes, já que a topologia pode ter um impacto significativo nos custos de capital. Neste contexto, os modelos analíticos podem ser usados em ferramentas de avaliação como critério de eliminação e previsão de custos, acelerando a tomada de decisões. Resultados experimentais revelam que a expressão proposta obtêm o valor da centralidade de intermediação com erro médio menor que* 3%.

1. Introdução

O planejamento de redes ópticas de telecomunicações envolve a identificação dos componentes necessários e a sua quantidade. A partir destas informações pode-se calcular o custo de capital (CapEx). Para o caso em que todas as informações sobre a rede estão disponíveis, incluindo a estrutura topológica e a matriz de tráfego, o planejamento torna-se uma tarefa relativamente fácil, já que existem ferramentas que auxiliam a sua execução. Contudo, na falta de alguma informação, os projetistas devem considerar aproximações para estimar quantidades e custos.

A estimação de quantidades e custos é de grande relevância para os operadores no momento da avaliação preliminar de soluções alternativas. Ela serve como ferramenta para tomadas de decisão [Silva 2010].Além da fase de planejamento inicial, as estimativas podem também ser usadas para prever o valor de parâmetros da rede sob diversas condições. É o caso da análise de tráfego multi-período. Nos últimos anos, vários pesquisadores apresentaram resultados sobre a estimação de variáveis de mérito em redes de telecomunicações. Em [Bodamer et al. 2004] os autores apresentam um método para estimar o número de transponders em redes multicamada. Em [Korotky 2004, Korotky 2005, Bhardwaj et al. 2005, Labourdette et al. 2005] os autores apresentam uma formulação semi-empírica para calcular um conjunto de variáveis de mérito, sem a necessidade de informações sobre a estrutura topológica da rede. Em [Pinto et al. 2009] é apresentado um modelo estatístico para o cálculo de custos de transmissão e gerenciamento de largura de banda. Em [Pavan et al. 2010] são identificadas, a partir de um conjunto de 29 topologias físicas de redes reais, características relevantes, como o número médio de saltos, coefiente de proteção e coeficiente de restauro, bem como aproximações para estas variáveis. Em [Routray et al. 2013] os autores apresentam uma expressão para estimar o comprimento médio dos enlaces, baseada apenas na informação sobre a área de cobertura da rede e do número de nós. Este modelo foi utilizado para estimar vários parâmetros de redes ópticas de telecomunicações em [Routray et al. 2014]. Estes modelos contribuem para a ampliação da caracterização de redes ópticas de telecomunicações e permitem maior precisão das previsões.

Neste artigo pretende-se estender o estudo realizado pelos autores apresentados acima, propondo uma aproximação para uma nova variável de mérito em topologias de redes de telecomunicações, a centralidade de intermediação, que mede a quantidade de vezes que um nó está presente nos caminhos mínimos entre os demais pares de nós. No âmbito das redes de telecomunicações esta medida pode representar a importância ou criticidade dos nós. Por exemplo, uma topologia pode ter a necessidade de reencaminhar um conjunto grande de canais em caso de falhas em um nó crítico. Numericamente, conhecida a topologia, a obtenção desta medida em grafos esparsos - como é o caso das redes de telecomunicações - tem complexidade $O(N^2 logN + NL)$ [Brandes 2001]. Na literatura há estudos que buscam a obtenção de algoritmos mais eficientes para a solução deste problema [Ausiello et al. 2013].

O artigo está organizado da seguinte forma: na Seção 2, é apresentado o conceito de centralidade de intermediação e também o conjunto de redes reais considerado neste estudo. Na Seção 3, é apresentado o método utilizado para a obtenção da expressão. A avaliação do método é apresentada na Seção 4. Por fim, na Seção 5, as considerações finais são apresentadas.

2. Centralidade de Intermediação

A centralidade de intermediação, BC (*Betweenness Centrality*), avalia a quantidade de vezes que um vértice (nó da topologia) está no caminho geodésico entre dois outros vértices. Ou seja, elevados índices de centralidade indicam que o vértice está em uma grande fração dos caminhos mais curtos entre os outros vértices, revelando a sua importância na topologia. De fato, a centralidade será dependente também da estrutura topológica. Para fins de análise preliminar de topologias alternativas, projetistas de redes podem optar por avaliar com mais detalhes as redes com menor centralidade de intermediação, já que este

resultado indica que os menores caminhos estarão mais distribuídos na topologia. Uma topologia com BC alto pode sofrer maior impacto (perda de tráfego) no caso de falhas.

Para um vértice v_k, a intermediação é indicada pelo fluxo entre um vértice fonte i e o seu destino j [Tizghafam and Leon-Garcia 2010].

A centralidade de intermediação de um vértice v_k em um grafo $G = (V, E)$, onde V é o número de vértices, e E o número de arestas (enlaces entre os nós) é dada pela expressão:

$$BC(v_k) = \sum_{i \neq j \neq v_k} \frac{b_{ij}(v_k)}{b_{ij}}, \tag{1}$$

onde b_{ij} representa o número total de caminhos mínimos entre i e j e $b_{ij}(v_k)$ representa o número de caminhos mínimos entre i e j que passam pelo vértice (v_k) [Brandes 2001]. O vértice com maior centralidade de intermediação será o mais central da rede [Ufimtsev and Bhowmick 2013, Freeman 1977].

Para analisar a centralidade de intermediação em topologias de redes reais de telecomunicações, considerou-se um conjunto com 41 topologias reais sobreviventes (com grafo 2-aresta-conexo). A Tabela 1 [Routray et al. 2013] apresenta o número de nós, N, enlaces, L, grau médio, $\langle\delta\rangle$ e a média ponderada da centralidade de intermediação, BC_{mp}.

Quando analisa-se topologias como a EON , Figura 1, observa-se que em algumas regiões há uma maior concentração de nós e enlaces. Em redes de telecomunicações, os nós com grau acima da média, são os principais candidatos a nó central da rede. Para

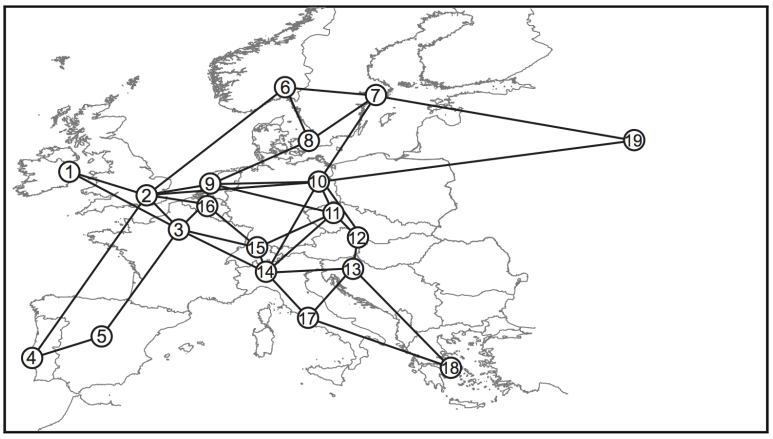
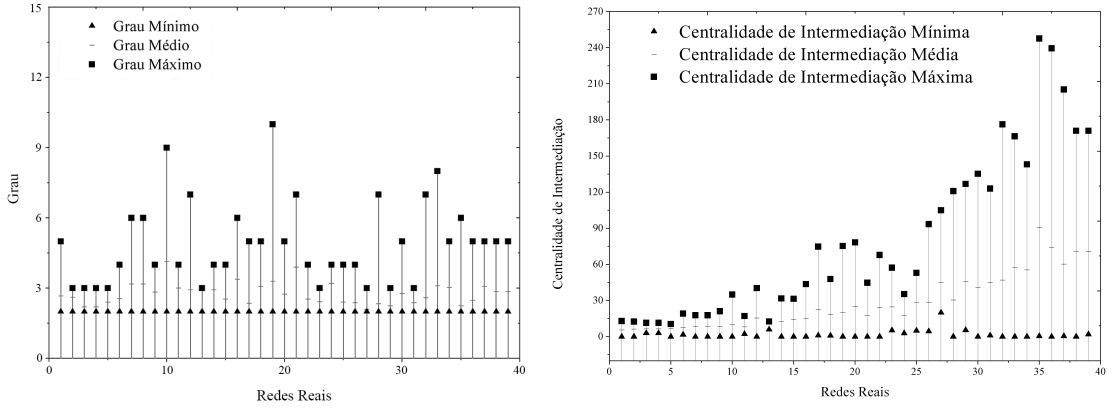


Figura 1. Topologia Física de Rede Óptica da Europa (EON) [Pavan et al. 2010].

a rede exemplo, o nó com índice 10 (de grau 7) é o nó mais central. Considerando o conjunto de redes de referência apresentado na Tabela 1, observou-se que 80% das redes apresentam esta característica, onde um nó com grau máximo (relativo aos demais nós da rede) é o nó mais central na rede. Por este motivo, considerou-se a média ponderada da centralidade de intermediação para representar a rede. A média ponderada é dada pela seguinte expressão:

$$BC_{mp} = \sum_{1 \leqslant v_k \leqslant N}^{N} \frac{BC(v_k)}{\delta(v_k)}, \tag{2}$$

onde N é o número total de nós, sendo BC a soma das centralidades de intermediação obtidas na Equação (1) para cada nó, divido pelo grau do mesmo. O valor da média ponderada da centralidade de intermediação BC_{mp} apresentado na coluna 6 da Tabela 1 foi obtido pela Equação(2). Na Figura 2(a) é apresentado o grau mínimo, médio e máximo de cada rede, e na Figura 2(b) é apresentada a centralidade mínima, média e máxima de cada rede.



(a) (b)

Figura 2. (a) valor Mínimo, Médio e Máximo para o grau nodal e, (b) centralidade de intermediação para 39 topologias. As topologias 40 e 41 (1) não são apresentadas no gráfico em função da escala, mas possuem centralidade de intermediação mínima de 0 e 13,5 , média de 212,22 e 334,17 e máxima de 631,42 e 1720,56 respectivamente, e grau mínimo 2 e 2, grau médio 2,18 e 3,42 e grau máximo 3 e 6.

3. Método Proposto para Obtenção da Centralidade de Intermediação

O método consiste em obter os coeficientes de uma função matemática, que relacione a média ponderada da centralidade em função do número de nós e do número de enlaces em cada nó, com base em dados de topologias reais. Utilizando o método dos mínimos quadrados para obter os coeficientes da função, avalia-se a correspondência entre os dados das topologias e a função obtida, através do coeficiente de correlação.

Pretende-se ajustar os coeficientes de uma função polinomial, utilizando os dados conhecidos da média ponderada da centralidade de intermediação $BC_{mp}(k) = f(N_k, L_k)$, para $k = 0, 1, \ldots, n$, para $n + 1$ pares de pontos de cada uma das k topologias, onde N_k e L_k são o número de nós e o número de enlaces de uma determinada topologia, respectivamente.

Dispondo os dados de média ponderada da centralidade (P) em função do número de nós (N) e o número de enlaces (L) em um gráfico cartesiano, foi observado que existe uma dependência levemente curva, com concavidade negativa em relação a L e fortemente positiva em relação a N. Esse tipo de dependência pode ser modelada com funções polinomiais, de grau 1 ou 2. Foram testados modelos polinomiais

Tabela 1. REDES REAIS DE REFERÊNCIA

Número	Rede	N	L	$\langle\delta\rangle$	BC_{mp}
1	Via Network	9	12	2,67	5,52
2	Abilinecore	10	13	2,60	6,32
3	BREN	10	11	2,20	6,50
4	Learn	10	11	2,20	6,50
5	RNP	10	12	2,40	6,58
6	Compuserv	11	14	2,55	7,42
7	Cesnet	12	19	3,17	8,35
8	Cesnet1	12	19	3,17	8,35
9	vBNS	12	17	2,83	8,55
10	Italy	14	29	4,14	10,30
11	NSF Network	14	21	3,00	8,04
12	Austria	15	22	2,93	15,47
13	Austria Aconet	15	22	2,93	10,50
14	Hibernia	15	23	2,93	12,81
15	Mzima	15	19	2,53	14,42
16	Garr-b Changed	16	27	3,38	15,13
17	ARNES	17	20	2,35	22,54
18	Germany	17	26	3,06	18,55
19	Redirisnet	17	28	3,29	20,28
20	CANARIE	19	26	2,74	25,29
21	EON	19	37	3,89	17,47
22	Memorex	19	24	2,53	24,12
23	NLR	19	23	2,42	24,64
24	ARPANET	20	32	3,20	17,31
25	Sweden	20	24	2,40	28,15
26	Pionier	21	25	2,38	28,56
27	Bulgaria	23	24	2,09	45,00
28	COX	24	40	3,33	30,34
29	Sanet	25	28	2,24	45,68
30	Portugal	26	36	2,77	40,84
31	New Network	26	31	2,38	44,70
32	Renater	27	35	2,59	47,05
33	CERNET	29	45	3,10	57,21
34	IBN 31	31	47	3,03	55,39
35	LONI	33	37	2,24	90,48
36	Metrona	33	41	2,48	74,33
37	COST 37	37	57	3,08	60,29
38	OMI1	38	54	2,84	70,97
39	Omicon	38	54	2,84	70,97
40	Internet 2	56	61	2,18	212,22
41	USA 100	100	171	3,42	334,17

$P(N, L) = a_0 + a_1 L^p + a_2 N^p$ considerando diferentes combinações de valores de p e q ($p = 1$, $p = 2$ e $q = 1$, $q = 2$). A combinação com menor coeficiente de correlação foi para $p = q = 2$ e a função é apresentada na Eq. (3).

$$P_2(N, L) = |a_0 + a_1 N^2 + a_2 L^2|, \tag{3}$$

onde a média ponderada da centralidade de intermediação é positiva para qualquer número de nós e enlaces em uma topologia, $P_2(N, L) > 0$, para qualquer $N > 0$ e $L > 0$.

O ajuste da Equação (3) é obtido fazendo com que a diferença dada pela Equação (4), seja mínima.

$$Q = \sum_{k=0}^{n} \Big[f(N_k, L_k) - |a_0 + a_1 N_k^2 + a_2 L_k^2| \Big]^2, \tag{4}$$

Substituíndo os $n + 1$ valores de $BC_{mp}(k) = f(N_k, L_k)$ na Equação (3) obtem-se o sistema linear dado pela Equação (5).

$$\begin{pmatrix} 1 & N_0^2 & L_0^2 \\ 1 & N_1^2 & L_1^2 \\ \vdots & \vdots & \vdots \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} BC_{mp}(0) \\ BCmp(1) \\ \vdots \\ BC_{mp}(n) \end{pmatrix}. \tag{5}$$

Sejam A, x e b, a matriz dos coeficientes, o vetor das incógnitas e o vetor dos termos independentes da Equação (5), respectivamente.

$$A * x = b \tag{6}$$

Multiplicando a Equação (6) pela matriz A^T, obtem-se um sistema linear conhecido como equações normais do sistema (6), que expandido toma a forma da Equação (7).

$$\begin{bmatrix} n & \sum N_k^2 & \sum L_k^2 \\ \sum N_k^2 & \sum L_k^2 & \sum L_k^2 N_k^2 \\ \sum N_k^2 & \sum N_k^2 L_k^2 & \sum L_k^4 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum BC_{mp}(k) \\ \sum N_k^2 BC_{mp}(k) \\ \sum L_k^2 BC_{mp}(k) \end{bmatrix}. \tag{7}$$

A solução do sistema da Equação (7), foi obtida neste trabalho pelo método de Gauss e dá os parâmetros ótimos, no sentido dos mínimos quadrados, da Equação (4) [Burden and Faires 2008].

A adequação do modelo pode ser feita comparando cada valor de $BC_{mp}(k)$ com o respectivo valor predito pelo modelo $P_2(N_k, L_k)$. Em uma análise ponto a ponto, a diferença entre esses dois valores é o resíduo [Barroso et al. 1987], entendido como parte do valor observado que o ajuste não foi capaz de modelar. Em uma análise mais geral, o coeficiente de correlação, calculado pela Equação (8), é mais usado para avaliar a qualidade do ajuste.

$$r_k = BC_{mp}(k) - \hat{BC}_{mp}(k), \tag{8}$$

onde, para nosso modelo, $\hat{BC}_{mp}(k)$ é dado pela equação (3). Os resíduos podem ser vistos como parte do valor observado que o ajuste não foi capaz de explicar.

Um método para medir a qualidade do ajuste linear é através do coeficiente de correlação:

$$R^2 = 1 - \sum_{k=1}^{n} \frac{(BC_{mp}(k) - \hat{BC}_{mp}(k))^2}{(BC_{mp}(k) - \frac{\sum BC_{mp}(k)}{n})^2}, \tag{9}$$

quanto mais próximo de 1 melhor é o ajuste, sendo que $0 < R^2 < 1$.

Aplicando o método proposto em dados das redes reais apresentadas na Tabela 1, foi obtido um ajuste com coeficiente de correlação $0,99$, ou seja, com um resíduo muito baixo. Foram encontrados os seguintes valores : $a0 = 4,236$, $a1 = 0,09$ e $a2 = -0,02$, a partir da Equação(7).

Seja um grafo conexo ou não com $n \geqslant 3$ nós. Onde v_k de G e $BC(v_k)$ a centralidade de intermediação de v_k. Então

$$0 \leqslant BC(v_k) \leqslant \frac{N^2 - 3N + 2}{2}, \tag{10}$$

onde o limite para a centralidade de intermediação de um grafo é alcançado se e somente se G for um grafo estrela S_n e v_k o vértice central [de Freitas 2010].

Sendo que os valores obtidos na Equação (3) estão dentro dos limites, tanto para caso máximo e o mínimo da Equação (10), entretanto as redes de telecomunicações nunca chegam a este limitante pois o grau de um nó v_k será sempre maior que um. A Figura 3 apresenta diagramas de dispersão com os valores da centralidade de intermediação em função do número de nós e enlaces. Sendo a centralidade de intermediação uma métrica muito bem aceita para determinar a sobrecarga de nó em uma rede qualquer [Tizghafam and Leon-Garcia 2010].

4. Experimentos e Resultados

Em [Girolimetto et al. 2013] é apresentada uma ferramenta livre para geração de topologias sobreviventes, *NTTGen*, com as características descritas em [Pavan et al. 2010]. O presente artigo utiliza a ferramenta com o intuito de validar a função matemática obtida a partir do conjunto de redes reais apresentado na Tabela 1.

As topologias geradas foram divididas em 5 subconjuntos, com os graus médios $2,5$, $3,0$, $3,5$, $4,0$ e $4,5$. Foram gerados 50 topologias sobreviventes com número de nós entre 10 e 100. Em seguida foi obtido os valores da média ponderada da centralidade de intermediação considerando a Equação (3) para cada topologia, aplicando assim a função:

$$BC_{est}(N, L) = |4,236 + 0,09N^2 - 0,02L^2|. \tag{11}$$

Para cada topologia correspondente ao conjunto gerado, através do valor obtido pela expressão BC_{est} (11) foi calculado o resíduo através da Equação (8), resultando em um coeficiente de correlação de $0,87$, Figura (4); com erro médio de $2,88\%$.

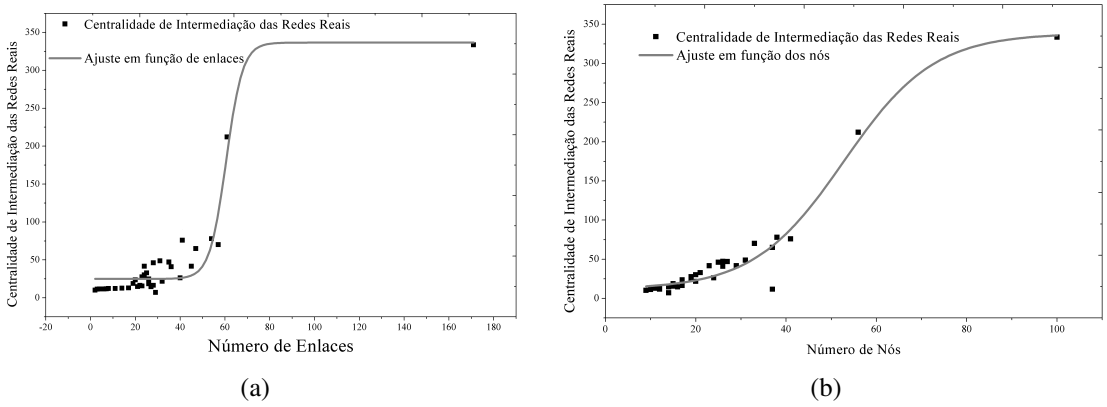


Figura 3. Centralidade de Intermediação em função do número de enlaces (a) determina o número de nós e, (b) está em função do número de nós, portanto determina o número de enlaces nas 41 topologias, sendo que o valor da centralidade de intermediação apresenta-se em ordem crescente.

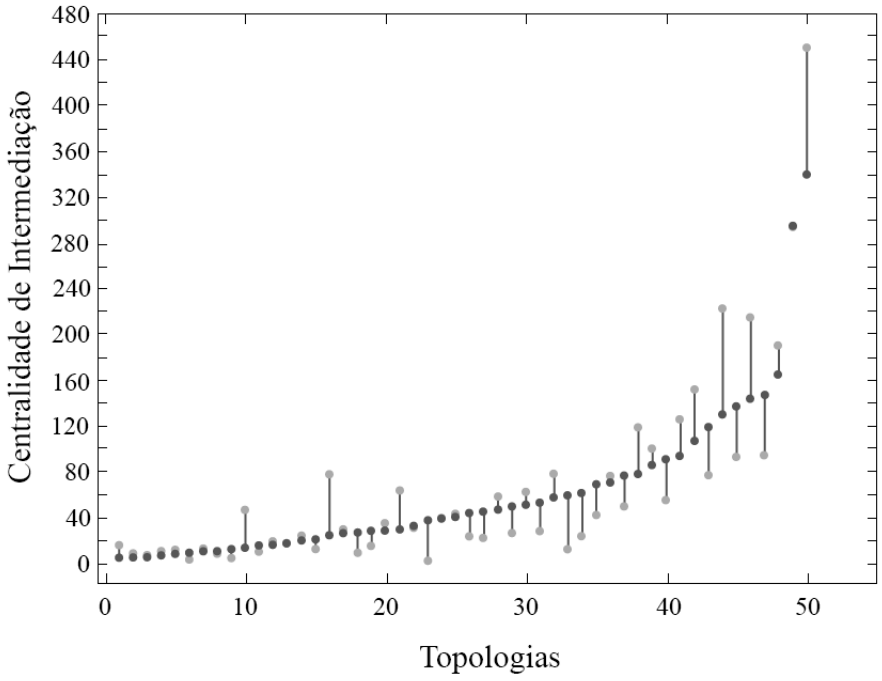


Figura 4. Topologias sobreviventes geradas pelo software *NTTGen*. Os pontos em cor cinza representam a centralidade de intermediação estimada pela função matemática e os pontos em cor preta os calculados numericamente (exatos).

5. Considerações Finais

A centralidade de intermediação é uma variável de mérito das redes de telecomunicações. Neste trabalho, foi proposta uma expressão matemática para estimar o valor da centralidade de intermediação de uma rede, sem a necessidade de conhecer a topologia. A expressão é resultado de um método de ajuste de curvas pelos mínimos quadrados, em que se obteve um ajuste com coeficiente de correlação de 0,99. A expressão de aproximação proposta, quando aplicada à redes de telecomunicações, geradas aleatoriamente, apresenta um erro médio de 3%.

6. Agradecimentos

Este trabalho foi realizado com auxílio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), através do projeto "Desenvolvimento de Uma Ferramenta Computacional para a Geração de Topologias de Redes de Telecomunicações", referente ao edital 308/UFFS/2014.

Referências

Ausiello, G., Firmani, D., e Laura, L. (2013). The (betweenness) centrality of critical nodes and network cores. *Wireless Communications and Mobile Computing Conference (IWCMC) - IEEE*, 9:90–95.

Barroso, L. C., Barroso, M. M. A., Filho, F. F. C., de Carvalho, M. L. B., e Maia, M. L. (1987). *Cálculo Numérico – Com Aplicações – 2 edição*.

Bhardwaj, M., McCaughan, L., Korotky, S. K., e Sanniee, I. (2005). Analytical description of shared restoration capacity for mesh networks. *OSA Journal of Optical Networking*, 4(3):130–141.

Bodamer, S., Spath, J., e Glingener, C. (2004). An efficient method to estimate transponder count in multi-layer transport networks. *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, 3:1780 – 1785.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177.

Burden, R. e Faires, J. (2008). *Análise Numérica – 8 edição*.

de Freitas, L. Q. (2010). Medidas de centralidade em grafos. Master's thesis, Universidade Federal do Rio de Janeiro.

Freeman, L. C. (1977). A set of centrality based on betweenness. *Sociometry*, 40:35–41.

Girolimetto, M., Gallupo, R. A., e Pavan, C. (2013). Uma ferramenta livre para a geração de topologias físicas de redes de telecomunicações. *Salão de Ferramentas do 31º SBRC - Simpósio de Redes de Computadores e Sistemas Distribuídos*, 2:1135–1142.

Korotky, S. K. (2004). Network global expectation model: A statistical formalism for quickly quantifying network needs and costs. *IEEE Journal of Lightwave Technology*, 22(3):703–722.

Korotky, S. K. (2005). Analysis and optimization of two-tier networks with application to optical transport backbones. *J. Opt. Netw.*, 4(1):45–63.

Labourdette, J. F., Bouillet, E., Ramamurthy, R., e Akyama, A. A. (2005). Fast approximate dimensioning and performance analysis of mesh optical networks. *IEEE/ACM Journal of Transactions on Networking*, 3(4):906–917.

Pavan, C., Morais, R. M., da Rocha, J. R. F., e Pinto, A. N. (2010). Generating realistic opticasl transport network topologies. *IEEE/OSA Journal of Optical Communications and Networking*, 2:80–90.

Pinto, A. N., Pavan, C., e Morais, R. M. (2009). A statistical model for capex fast calculation in optical transport networks. In *Proceedings of the 11th International Conference on Transparent Optical Networks, ICTON'09*, volume 11, páginas 1–4, Island of São Miguel, Azores, Portugal. IEEE.

Routray, S. K., Morais, R. M., Ferreira, J. R., e Pinto, A. N. (2013). Statistical model for link lengths in optical transport networks. *Optical Communications and Networking, IEEE/OSA Journal of*, 5(7):762–773.

Routray, S. K., Sahin, G., Ferreira, J. R., e Pinto, A. N. (2014). Estimation of link-dependent parameters of optical transport networks from the statistical models. *Optical Communications and Networking, IEEE/OSA Journal of*, 6(7):601–609.

Silva, T. S. A. (2010). Um estudo de medidas de centralidade e confibilidade em redes. Master's thesis, Centro Federal de Educação Técnologica Celso Suckow da Fonseca.

Tizghafam, A. e Leon-Garcia, A. (2010). Betweenness centrality and ristance - distance in communication networks. *IEEE Network*, 24:10–16.

Ufimtsev, V. e Bhowmick, A. (2013). Identifying high betweenness centrality vertices in large noisy networks. *IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum*, 27:2234–2237.

Plataforma para Monitoramento de Métricas de Nível de Serviço em Redes Definidas por Software

Pedro H. A. Rezende, Paulo R. S. L. Coelho, Luís F. Faina, Lasaro Camargos, Rafael Pasquini

Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU)

pedroh@mestrado.ufu.br

{paulocoelho, faina, lasaro, rafael.pasquini}@ufu.br

Resumo. *Este artigo apresenta e avalia uma plataforma para monitoramento de métricas de nível de serviço em Redes Definidas por Software (SDN) OpenFlow, denominada SDNMon. Por meio de monitoramentos frequentes e em diferentes granularidades, incluindo fluxos individuais, a SDNMon mantém medições precisas da vazão e atraso experimentados. As avaliações incluem duas formas de levantamento de dados, uma proposta baseada em consultas explícitas (polling) a contadores OpenFlow mantidos pelos elementos de rede e uma alternativa baseada em amostragens feitas com o protocolo sFlow.*

Abstract. *This paper presents and evaluates a platform for monitoring service-level metrics in OpenFlow Software Defined Networks (SDN), named SDNMon. The platform supports frequent network observation, at different granularity levels, including per-flow observation, being able to present accurate on-the-fly statistics regarding throughput and delay. The evaluation considered two statistical data gathering approaches, a proposed polling-based mechanism which collects information kept by OpenFlow counters at the network elements, and an alternate sampling-based mechanism which uses sFlow.*

1. Introdução

A utilização de um plano de controle desacoplado dos elementos de rede, conforme promovido no cenário de Redes Definidas por Software (SDN), estabelece um rico cenário para pesquisas em diversas áreas. A implementação deste plano de controle ocorre através de pelo menos um controlador, capaz de atuar em todos os elementos de rede presentes no plano de dados, programando seu comportamento de encaminhamento de tráfego. Esta atuação por parte do controlador é geralmente baseada em uma visão completa da rede, e utiliza métricas convencionais para alimentar algoritmos de caminho mais curto, tais como o número de saltos [Kim and Feamster 2013].

Embora muitas sejam as pesquisas relacionadas ao plano de controle SDN [Kreutz et al. 2014, Guedes et al. 2012, dos Passos Silva et al. 2013], apenas recentemente trabalhos investigam o monitoramento do plano de dados em SDN. Ademais, os cenários explorados são bem específicos, tais como, monitoramento de transmissões TCP [Suh et al. 2014], monitoramento dos elementos de entrada e saída da rede [van Adrichem et al. 2014] e monitoramento de latência utilizando sondas (*probes*) em cada enlace [Phemius and Bouet 2013].

A principal contribuição deste artigo é uma plataforma de monitoramento, denominada SDNMon, que provê um arcabouço para a implementação de diferentes técnicas de monitoramento, e portanto mais geral que os trabalhos anteriores. A SDNMon é independente da aplicação sendo transportada na rede e capaz de produzir uma visão detalhada do plano de dados, em diferentes granularidades. Além de unificar e melhorar o conjunto de ferramentas disponíveis aos administradores de rede, a SDNMon pode atuar em conjunto com outros módulos do controlador, por exemplo, enriquecendo a visão sobre o plano de dados, permitindo a definição de rotas que consideram a maximização da taxa de transmissão ou a minimização do atraso.

Como uma segunda contribuição deste trabalho, é proposto um mecanismo de monitoramento baseado em consultas explícitas (*polling*) aos contadores mantidos em cada um dos elementos de rede, definidos na especificação do OpenFlow [ONF 2013]. Basicamente, o mecanismo de monitoramento proposto baseado em *polling* é implementado no arcabouço promovido pela plataforma SDNMon. O mecanismo de *polling* é capaz de adaptar-se a situação de carga da rede, produzindo uma visão detalhada em diferentes níveis de granularidade, incluindo a condição dos elementos de rede presentes no plano de dados, das portas que os compõem, das aplicações utilizando a rede e dos fluxos individuais sendo encaminhados na SDN.

Para validar a SDNMon, a seção de experimentos contempla medições efetuadas utilizando duas diferentes abordagens de monitoramento: a primeira é o mecanismo proposto baseado em *polling*; a segunda utiliza o protocolo sFlow [sFlow 2014, Phaal et al. 2001], efetuando amostragens de pacotes sendo transmitidos na rede. Os experimentos apresentados neste artigo investigam o funcionamento de ambas as abordagens e, também, avaliam o monitoramento considerando três aplicações legadas, incluindo transmissões UDP feitas a partir do gerador de tráfego *Iperf* [Iperf 2014], transmissões TCP feitas utilizando a aplicação *scp* para a cópia de um arquivo, e *streaming* de vídeo utilizando o VLC [VLC 2014].

As demais seções estão organizadas da seguinte forma: A Seção 2 introduz os principais trabalhos de monitoramento presentes na literatura, oferecendo uma breve comparação com a plataforma proposta. A Seção 3 detalha a plataforma SDNMon e o mecanismo de monitoramento proposto baseado em *polling*. A Seção 4 apresenta os resultados das avaliações. A Seção 5 conclui o artigo e apresenta os trabalhos futuros.

2. Trabalhos Relacionados

Em [van Adrichem et al. 2014], os autores utilizam um mecanismo de *polling* para coletar dados referentes aos fluxos, permitindo apresentar medidas de largura de banda consumida por fluxo e da taxa de pacotes perdidos. Ainda, utilizando um mecanismo de *push*, os autores conseguem analisar o atraso fim-a-fim experimentado. O principal diferencial entre o trabalho apresentado em [van Adrichem et al. 2014] e a plataforma SDNMon está na quantidade de detalhes capturados pelo módulo de monitoramento baseado em *polling*. Em [van Adrichem et al. 2014], leituras são executadas apenas nos elementos de rede de ingresso e egresso, enquanto na SDNMon as leituras são feitas em todos os elementos de rede que compõem o plano de dados, permitindo o monitoramento dos fluxos em cada um dos enlaces por onde são encaminhados.

O protocolo sFlow [sFlow 2014, Phaal et al. 2001] trabalha na metodologia de

push, utilizando agentes presentes em cada um dos elementos de rede, responsáveis por amostrar pacotes e enviá-los para um nó central denominado coletor. As amostragens do sFlow são realizadas apenas na granularidade das portas dos elementos de rede, através da configuração de uma taxa de amostragem N. Por exemplo, utilizando uma taxa de amostragem $N = 100$, o agente sFlow irá amostrar um pacote a cada cem pacotes transmitidos em determinada porta. O coletor recebe uma cópia dos cabeçalhos dos pacotes amostrados, permitindo estimar o total de bytes transmitidos e a taxa de transmissão.

O sFlow é modificado em [Suh et al. 2014] para inspecionar o número de sequência do TCP presente nos pacotes, como forma de aumentar a precisão das amostragens. De fato, o mecanismo apresentado em [Suh et al. 2014] melhora o funcionamento do sFlow, porém restringe as amostragens apenas a transmissões TCP. O principal diferencial do sFlow, ou mesmo da modificação publicada em [Suh et al. 2014], frente ao mecanismo de *polling* proposto neste artigo, refere-se a atuação em diferentes níveis de granularidade (portas, filas, fluxos, elementos de rede), na metodologia de *pull* para coletar informações mantidas pelos contadores previstos no OpenFlow.

Em [Phemius and Bouet 2013], os autores utilizam mensagens OpenFlow como forma de medir a latência experimentada em cada enlace. Basicamente, propõem a injeção de mensagens via controlador em um *switch* da malha de elementos de rede e a posterior recuperação destas mensagens em um *switch* adjacente. Através do monitoramento do tempo necessário ao retorno das mensagens ao controlador, os autores apresentam os valores de latência. Na SDNMon não são utilizadas mensagens injetadas pelo controlador; as medições são feitas considerando os dados pertencentes aos fluxos monitorados, comparando o volume de dados encaminhado em cada um dos *switches*.

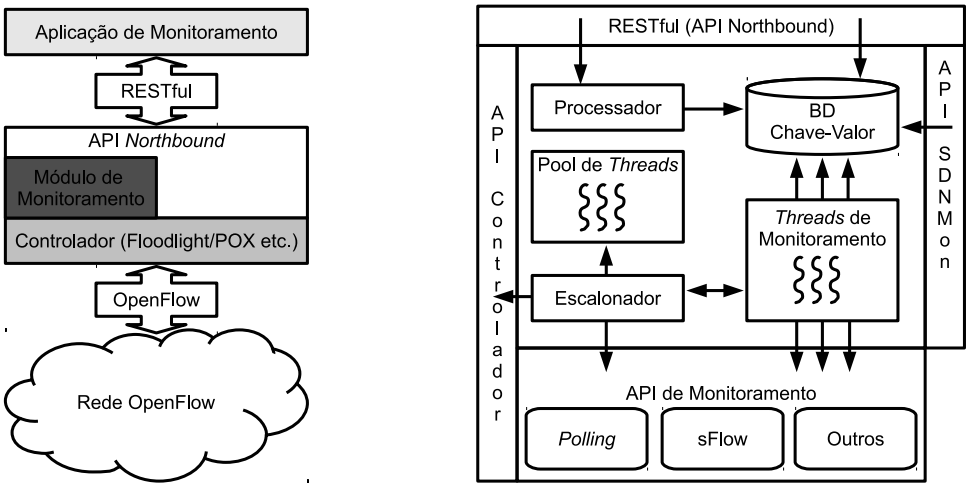
3. Plataforma de Monitoramento SDNMon

Conforme ilustra a Figura 1(a), o desenvolvimento da plataforma de monitoramento SDNMon é feito sob o cenário de SDN, composta por elementos de rede OpenFlow, sendo os elementos de rede responsáveis pela composição do plano de dados e ao menos um controlador responsável pela composição do plano de controle.

A SDNMon é desenvolvida como uma extensão ao controlador, utilizando informações presentes neste para auxiliar no monitoramento, tais como informações sobre a topologia do plano de dados. Todas as informações coletadas pelo módulo de monitoramento estão disponíveis utilizando o protocolo RESTful, através de uma extensão à API *northbound* do controlador. Nesta interface, a Aplicação de Monitoramento consegue interagir com o Módulo de Monitoramento, obtendo dados para apresentação em uma interface gráfica e, também, indicando qual o nível de granularidade desejado nas ações de monitoramento do mecanismo de *polling*.

A Figura 1(b) detalha a arquitetura do Módulo de Monitoramento proposto na SDNMon, cuja característica principal é a adaptabilidade ao cenário, em reação a variações no volume de tráfego na rede, baseada na utilização de um conceito de *pool* de *threads*. A Seção 3.1 detalha esta característica de adaptabilidade da SDNMon, implementada pelo mecanismo de *polling* proposto.

O Escalonador utiliza informações referentes à topologia do plano de dados obtidas no controlador (através da API do Controlador) para instanciar o módulo de monitoramento, provendo inicialmente uma *thread* para monitorar integralmente cada um dos



(a) Cenário considerado.

(b) Arquitetura do Módulo de Monitoramento.

Figura 1. Plataforma SDNMon.

elementos de rede. As ações de monitoramento ocorrem através da API de Monitoramento, que pode contemplar diferentes abordagens. A Figura 1(b) apresenta as opções *Polling* e sFlow, utilizadas nas experimentações deste artigo, mas também seria possível utilizar outras abordagens, como a proposta em [Suh et al. 2014] descrita na Seção 2.

Todos os dados coletados são armazenados na Base de Dados do tipo chave-valor pelas *threads*. A chave refere-se a um elemento de rede, uma porta, uma fila ou a um fluxo. No caso de um elemento de rede, das portas ou das filas, o campo valor armazena dados referentes ao volume agregado de tráfego sendo experimentado neste componente. No caso de um fluxo, o valor do objeto é o conjunto de arestas do caminho percorrido pelo fluxo; para cada aresta há uma outra entrada na base de dados, cuja chave associa o fluxo ao par de *switches* que compõe a aresta e o valor é o conjunto de informações relativas à mesma. Desta forma, as *threads* podem atualizar as arestas concorrentemente.

Especificamente no caso dos monitoramentos utilizando o mecanismo proposto de *polling*, os dados coletados estão em conformidade com os contadores previstos na especificação do OpenFlow [ONF 2013]. Alguns dos dados não precisam de nenhum tipo de processamento para serem disponibilizados para a Aplicação de Monitoramento, tais como informações sobre a capacidade dos enlaces e o número de portas dos elementos de rede. Porém, alguns dos valores precisam de um grau variado de processamento para serem apresentados. Desta forma, o módulo de monitoramento contempla um Processador, responsável pelos cálculos. As Seções 3.2 e 3.3 descrevem o uso do processador.

Os dados referentes à taxa de transmissão obtidos pelo sFlow são pré-processados pelo coletor do sFlow, utilizando as amostras enviadas a ele pelos agentes. Neste caso, a opção sFlow implementada pelo módulo de monitoramento recupera estes dados junto ao coletor e armazena na base de dados. O Processador também é utilizado para adaptar os dados ao formato adequado para exibição na Aplicação de Monitoramento.

Finalmente, é importante destacar que a plataforma de monitoramento SDNMon é o primeiro componente na direção de uma plataforma para o provisionamento de quali-

dade de serviço que encontra-se em desenvolvimento. Através da API SDNMon ilustrada na Figura 1(b), os módulos da plataforma de qualidade de serviço podem obter os dados armazenados na base de dados de monitoria, observando se acordos de nível de serviço estão sendo honrados. A plataforma de qualidade de serviço está fora do escopo deste trabalho e será detalhada em um trabalho futuro.

3.1. Adaptabilidade do *Pool* de *Threads* no Mecanismo de *Polling*

Conforme mencionado anteriormente, o Escalonador consulta informações topológicas através da API do Controlador e instancia uma *thread* de monitoramento para cada um dos elementos de rede compondo a SDN. Na sequência, a partir das informações coletadas via mecanismo de *polling* em cada um dos elementos de rede, as *threads* de monitoramento realimentam o escalonador, levando a adaptações no número de *threads* instanciadas.

Os dados são coletados através do envio de mensagens de *Statistic Request* a partir de cada uma das *threads* aos respectivos elementos de rede. O teor das solicitações enviadas nas mensagens é definido pelo escalonador de acordo com o escopo de monitoramento atribuído a cada uma das *threads*. Uma vez que inicialmente há uma *thread* para cada elemento de rede, as mensagens utilizam coringas (*wildcards*) em todos os campos previstos na estrutura da mensagem de *Statistic Request*. Ao receber uma requisição como esta, o elemento de rede devolve uma mensagem do tipo *Statistic Reply*, contendo todos os valores contabilizados.

Ao receber o retorno de um elemento de rede, a *thread*, além de armazenar as informações na base de dados, analisa o volume de informações retornado e comunica ao escalonador para que decida sobre a necessidade de novas *threads* para eliminar gargalos de monitoramento. Utilizando desta técnica de *feedback*, o escalonador é capaz de instanciar novas *threads* e definir o escopo de monitoramento atuando nos campos solicitados nas mensagens de *Statistic Request*, ou seja, o escalonador determina o escopo de monitoramento através da definição dos coringas em cada uma das *threads* instanciadas.

A partir da especificação atual dos coringas presente no OpenFlow, a SDNMon consegue instanciar *threads* responsáveis pelo monitoramento na granularidade de elementos de rede, portas individuais dos elementos de rede, filas ou fluxos individuais. Como investigação futura, um dos itens de interesse é o suporte por parte do OpenFlow para a definição de coringas que representam intervalos, permitindo a instanciação de *threads* responsáveis, por exemplo, pelo monitoramento de um conjunto de portas dos elementos de rede ou um conjunto de filas. Também investigaremos aspectos relacionados ao sincronismo entre as *threads*, e o impacto que o sincronismo causa no volume de tráfego de controle e a sobrecarga de processamento aplicada nos elementos compondo o plano de dados da SDN.

3.2. Calculando a Taxa de Transmissão

A Figura 2 apresenta o método proposto neste trabalho para o cálculo da taxa de transmissão a partir do total de dados transmitidos. O método pode ser aplicado independentemente da granularidade dos dados obtidos pelo mecanismo de *polling*. Considerando a coleta de dados referente a um fluxo em um mesmo elemento de rede, temos que a Série 1, apresentada na Figura 2, corresponde ao total de dados transmitidos no fluxo em questão para este elemento de rede. Através da coleta de dois pontos consecutivos é possível obter a taxa de transmissão referente ao fluxo neste elemento de rede.

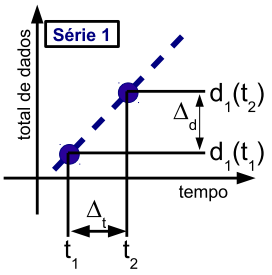


Figura 2. Cálculo da taxa de transmissão.

Conforme ilustrado na Figura 2, é possível obter o total de dados transmitidos entre os dois pontos coletados utilizando a fórmula $\Delta_d = d_1(t_2) - d_1(t_1)$, onde $d_x(t_y)$ representa o total de dados transmitidos na Série x em um determinado instante de tempo y. Ainda considerando estes dois pontos, é possível obter o intervalo de tempo entre eles, utilizando a fórmula $\Delta_t = t_2 - t_1$, onde t_y corresponde ao instante no qual as respectivas medições foram efetuadas. A partir destes valores, é possível obter a taxa de transmissão $T^1_{t1,t2} = \Delta_d/\Delta_t$ da Série 1 no intervalo t_1, t_2.

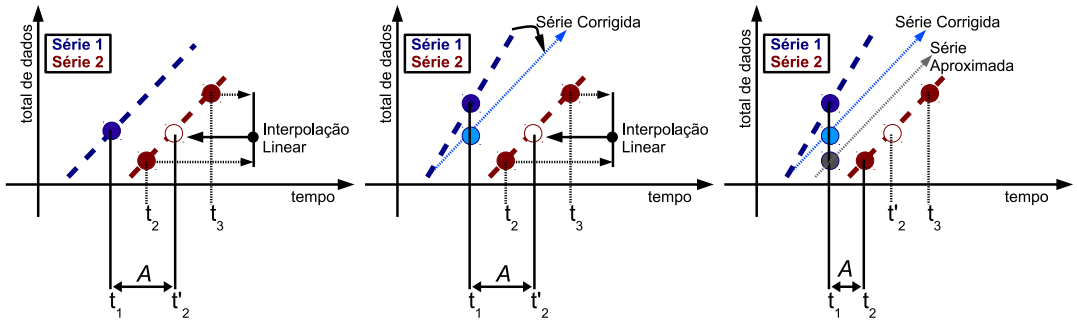
3.3. Calculando o Atraso

É comum encontrarmos trabalhos na literatura, por exemplo [Phemius and Bouet 2013] apresentado na Seção 2, que injetam sondas (pacotes de *probe*) na rede para determinar o atraso sendo experimentado. Este tipo de abordagem é bastante simples de ser implementada, mas pode não representar precisamente o atraso sendo experimentado por um determinado fluxo. Atualmente, é comum utilizarmos diversas filas de encaminhamento em uma única porta de um elemento de rede, fato este que exigiria a inserção de diversas sondas, específicas para cada uma das possíveis filas ao longo do caminho.

Em nossa abordagem, propomos a utilização de informações referentes ao total de dados transmitidos em cada um dos elementos de rede, de tal forma a efetuar o cálculo de atraso em diferentes granularidades, de acordo com os valores obtidos nos contadores especificados pelo OpenFlow. A Figura 3(a) apresenta o método proposto em um cenário livre de interferências, sem a ocorrência de descartes de dados, consistindo essencialmente em estimar o intervalo de tempo necessário para que cada elemento receba todos os dados enviados pelo anterior.

Considerando o cálculo de atraso para um determinado fluxo, a Figura 3(a) apresenta as Séries 1 e 2, coletadas em dois elementos de rede adjacentes, através dos quais o fluxo em questão é encaminhado. O método consiste em determinar t_1, t_2 e t_3, tal que $d_2(t_2) \leq d_1(t_1) \leq d_2(t_3)$. O próximo passo é determinar $T^2_{t2,t3}$, conforme definido na Seção 3.2. Finalmente, determinar t'_2, tal que $d_1(t_1) = d_2(t'_2)$, utilizando uma interpolação linear, para obter o atraso $A = t'_2 - t_1$.

O contador de total de bytes transmitidos do OpenFlow, mantido pelos elementos de rede, contabiliza o total bruto de bytes transmitidos, incluindo os bytes referentes aos pacotes efetivamente transmitidos e, também, os bytes referentes aos pacotes descartados. Desta forma, a Figura 3(b) ilustra como esta condição interfere no cálculo do atraso experimentado. Para determinar o atraso real, é preciso subtrair da Série 1 o total de dados descartados, obtendo o total de dados efetivamente transmitidos no instante de tempo em



(a) Séries sem a ocorrência de descarte de dados.

(b) Utilizando um contador de total de dados descartados.

(c) Sem a utilização de um contador de dados descartados.

Figura 3. Cálculo do atraso.

questão, representado pela Série Corrigida na Figura 3(b). Após a correção da série de dados, aplica-se o mesmo método descrito anteriormente para obter o valor de atraso.

Entretanto, a especificação do OpenFlow contempla apenas o contador referente aos bytes descartados por porta, não sendo possível obter informações referentes aos bytes descartados por fila ou fluxo. Sendo assim, a Figura 3(c) apresenta um cenário alternativo, onde o ponto da Série 1 obtido no instante t_1 é aproximado para o correspondente em total de dados do ponto da Série 2 obtido no instante t_2, considerando $d_{ap}(t_1) \approx d_2(t_2)$. Neste caso, descarta-se a interpolação linear, sendo possível obter o atraso $A = t_2 - t_1$. Entretanto, como pode ser observando na Figura 3(c), esta metodologia compromete a precisão do cálculo de atraso. A Seção 4 apresenta uma análise sobre o nível de precisão perdido neste caso.

Em um trabalho futuro, estenderemos uma implementação do OpenFlow, incluindo contadores referentes aos dados descartados em todos os níveis de granularidade (portas, filas e fluxos), em termos de total de pacotes descartados e total de bytes descartados. A inserção destes novos contadores permitiria análises mais precisas sobre o funcionamento da rede.

4. Experimentos

A Figura 4 apresenta o cenário utilizado para as análises da plataforma SDNMon. Foram consideradas a comunicação entre dois hospedeiros (H1 e H2) através de uma rede OpenFlow composta por cinco elementos de rede (SW1, SW2, SW3, SW4 e SW5). A infraestrutura de rede foi instanciada no Mininet versão 2.1.0, executando o Open vSwitch versão 2.0.2 nos elementos de rede e o controlador utilizado para a implementação do módulo de monitoramento é o Floodlight versão 0.90. Todos os experimentos foram efetuados em um Intel Core i3-2310M CPU 2.10GHz x 4 com 4 GB de memória RAM e sistema operacional Ubuntu 12.04.

Conforme mencionado anteriormente, o módulo de monitoramento da SDNMon foi implementado contando com dois mecanismos de monitoramento. O primeiro é o mecanismo proposto de *polling*, que efetua a coleta de valores armazenados pelos contadores previstos na especificação OpenFlow, através de consultas explícitas usando as mensagens de *Statistic Request* e *Statistic Reply*. O segundo mecanismo utiliza o protocolo sFlow, baseado em agentes executados nos elementos de rede que enviam as amostragens a um

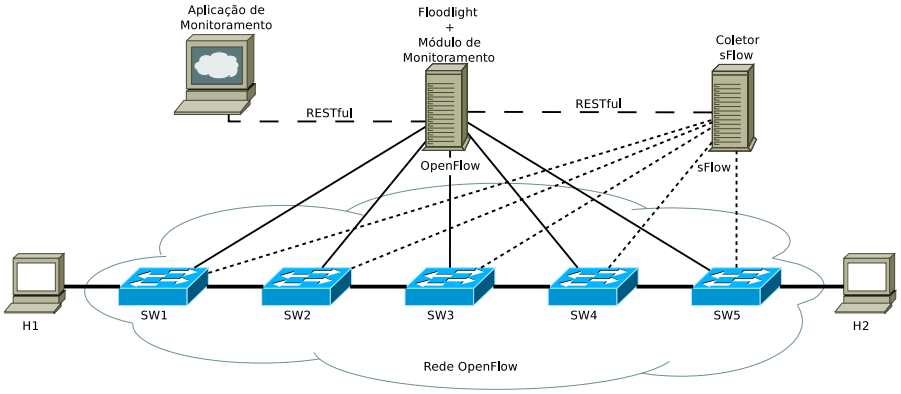
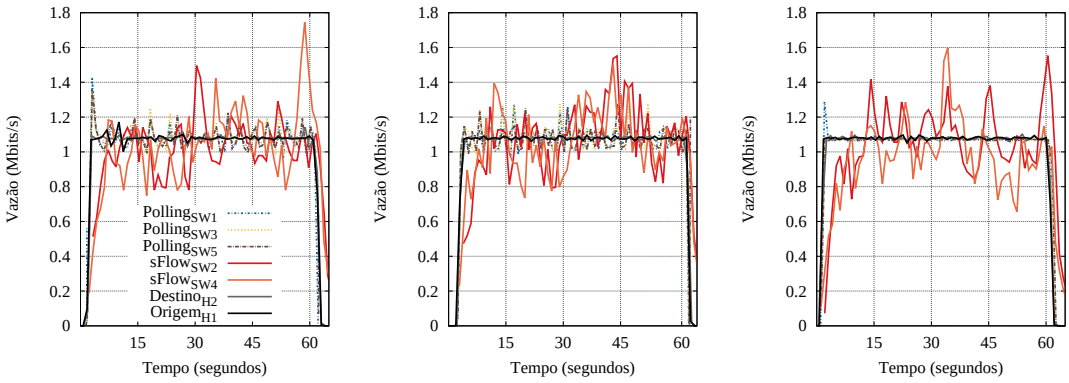


Figura 4. Cenário utilizado nas avaliações.

coletor central. As amostragens consideram um pacote dentro de um conjunto N de pacotes transmitidos, na granularidade das portas. O módulo de monitoramento comunica-se com o protocolo na versão sFlow-RT via RESTful.

4.1. Análise do Mecanismo de *Polling*

A Figura 5 apresenta uma análise referente à variações na taxa de coletas do módulo de monitoramento no caso do mecanismo de *polling*, objeto principal da análise efetuada nesta seção. O fluxo analisado foi gerado utilizando o *Iperf*, transmitindo UDP à taxa de 1 Mbps durante aproximadamente um minuto e utilizando pacotes com MTU de 1500 bytes. Para permitir uma visão completa sobre o mecanismo de monitoramento em todos os elementos de rede e permitir uma análise comparativa, são apresentados os dados de monitoramento do mecanismo de *polling* coletados nos elementos de rede SW1, SW3 e SW5 e os dados referentes ao sFlow coletados nos elementos de rede SW2 e SW4. Estes dados do mecanismo de monitoramento são comparados à transmissão real observada diretamente nas placas de rede dos hospedeiros H1 e H2, sendo H1 o hospedeiro de origem do tráfego analisado.



(a) 10 milissegundos. (b) 100 milissegundos. (c) 1 segundo.

Figura 5. Variação de intervalos de coleta do mecanismo de *polling*.

A metodologia foi variar a frequência entre leituras, considerando 10 milissegundos (Figura 5(a)), 100 milissegundos (Figura 5(b)), 1 segundo (Figura 5(c)) e armazenando apenas os valores observados quando havia mudança nos contadores. Em todas

as amostragens da Figura 5, o mecanismo sFlow efetuou a amostragem considerando $N = 10$, ou seja, eram considerados para amostragem um pacote a cada dez transmitidos.

A Tabela 1 apresenta os dados consolidados referentes à média e o desvio das análises apresentadas na Figura 5. Os valores apresentados na Tabela 1 consideram as observações feitas em todos os elementos de rede (SW1 a SW5) para ambas as metodologias de *polling* e sFlow. Como pode ser observado, o mecanismo de *polling* apresenta maior precisão em todos os cenários avaliados, apresentando sempre média próxima aos valores de H1 e H2 com um baixo desvio padrão.

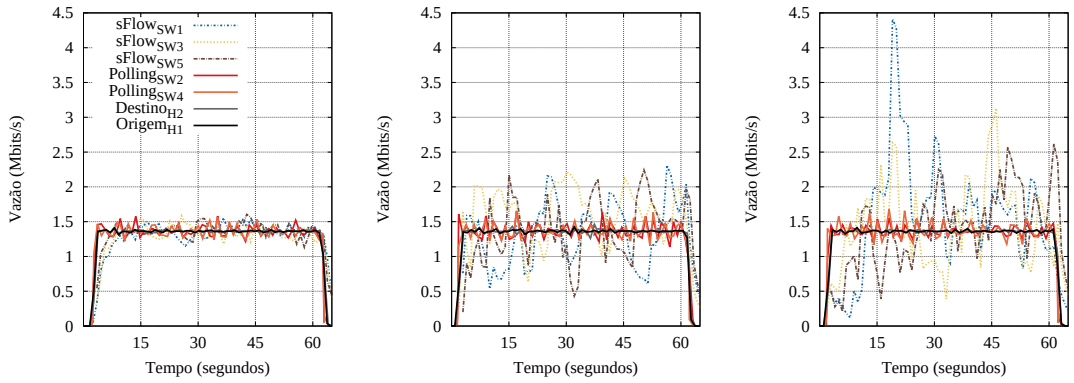
Tabela 1. Dados consolidados referentes à análise do mecanismo de *polling*.

Análise	Média e Desvio Padrão (em Mbits/s)			
	$Origem_{H1}$	$Destino_{H2}$	$Polling_{SW1..5}$	$sFlow_{SW1..5}$
Fig. 5(a) (10ms)	1,073 +/- 0,044	1,075 +/- 0,029	1,086 +/- 0,069	1,055 +/- 0,191
Fig. 5(b) (100ms)	1,072 +/- 0,048	1,070 +/- 0,041	1,065 +/- 0,155	1,081 +/- 0,176
Fig. 5(c) (1s)	1,072 +/- 0,048	1,068 +/- 0,057	1,068 +/- 0,029	1,060 +/- 0,170

4.2. Análise do Mecanismo de Amostragem sFlow

A Figura 6 analisa variações na taxa N de amostragem do sFlow. Por se tratar de um mecanismo de amostragem, o principal problema desta abordagem é ser utilizada em situações onde os pacotes amostrados possuem tamanho variado, uma característica comum nas redes de comutação por pacotes. Desta forma, a metodologia aplicada nestes testes foi a geração de 15 fluxos UDP entre H1 e H2 a uma taxa de 100 kbps. Cada fluxo possui MTU distinta, variando entre 100 e 1500 bytes, em incrementos de 100 bytes.

Neste caso, como o sFlow é o objeto principal de avaliação, são apresentados os dados de monitoramento da amostragem sFlow referentes aos elementos de rede SW1, SW3 e SW5 e, para comparação, são apresentados os dados referentes ao mecanismo de *polling* coletados nos elementos de rede SW2 e SW4. Em todas as análises da Figura 6, o mecanismo de *polling* efetuou medições em intervalos de um segundo.



(a) $N = 10$. (b) $N = 100$. (c) $N = 256$.

Figura 6. Variação da taxa de amostragem N do sFlow.

Como pode ser observado nos gráficos da Figura 6, mudanças na taxa de amostragem N comprometem a precisão do protocolo sFlow no caso de transmissões com pacotes

de tamanho variado. Além disso, a taxa de amostragem, conforme mencionado, é associada à porta dos elementos de rede, não possuindo caráter adaptativo, ou seja, não reage à mudanças no volume de tráfego. Por exemplo, uma taxa de amostragem $N = 256$ é o valor recomendado para portas de 10 Mbps, $N = 512$ para 100 Mbps, $N = 1024$ para 1 Gbps, $N = 2048$ para 10 Gbps e $N = 8192$ para 100 Gbps [sFlow 2014]. Embora o valor de amostragem da Figura 6(c) seja o recomendado para portas de 10 Mbps, a incidência de um volume inferior de tráfego também ocasiona perda de precisão.

Considerando as características dos fluxos UDP transmitidos, temos uma taxa de geração superior a quatrocentos pacotes por segundo, levando o agente sFlow a amostrar mais de quarenta pacotes por segundo no cenário com $N = 10$. Neste mesmo cenário, a coleta de dados dos contadores no mecanismo de *polling* está ocorrendo à frequência de uma coleta por segundo. Nestes experimentos, o mecanismo de *polling* está atuando na granularidade da porta de rede, da mesma forma como o sFlow atua. Pode-se ajustar o mecanismo de *polling* para atuar na granularidade de fluxos, exibindo dados individuais referentes a cada um dos quinze fluxos do experimento, uma característica não possível de ser implementada pelo sFlow.

A Tabela 2 apresenta a consolidação dos dados referentes ao sFlow. Especificamente sobre os resultados apresentados na Figura 6(a), é possível notar que a precisão do sFlow está bem próxima aos valores apresentados pelo mecanismo de *polling*, mas ainda assim indicando vantagem do mecanismo de *polling*.

Tabela 2. Dados consolidados referentes à análise do sFlow.

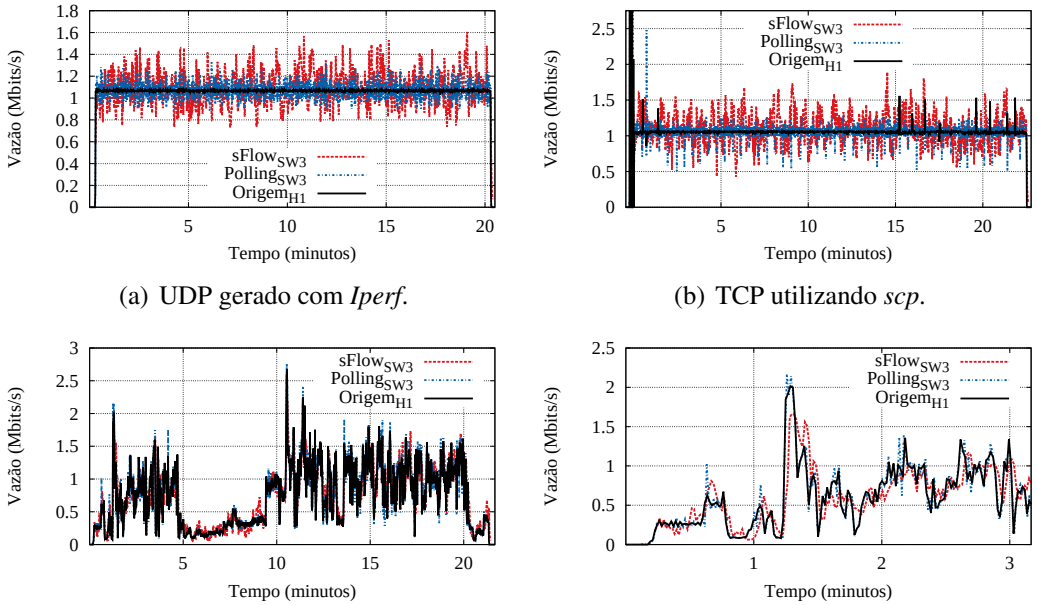
Análise	Média e Desvio Padrão (em Mbits/s)			
	$Origem_{H1}$	$Destino_{H2}$	$Polling_{SW1..5}$	$sFlow_{SW1..5}$
Fig. 6(a) ($N = 10$)	1,356 +/- 0,045	1,352 +/- 0,072	1,359 +/- 0,109	1,327 +/- 0,185
Fig. 6(b) ($N = 100$)	1,360 +/- 0,023	1,362 +/- 0,027	1,360 +/- 0,123	1,431 +/- 0,436
Fig. 6(c) ($N = 256$)	1,354 +/- 0,064	1,357 +/- 0,042	1,357 +/- 0,122	1,493 +/- 0,636

4.3. Análise da Taxa de Transmissão em Aplicações Legadas

A Figura 7 apresenta uma análise com três aplicações legadas, incluindo transmissão UDP feita a partir do gerador de tráfego *Iperf* [Iperf 2014] à taxa de 1 Mbps, transmissão TCP feita utilizando a aplicação *scp* limitada a vazão de 1Mbps durante a cópia de um arquivo, e transmissão de vídeo utilizando o VLC [VLC 2014] sem qualquer controle referente à taxa de transmissão. Em todas as três aplicações legadas avaliadas, o período de análise considera um intervalo aproximado de vinte minutos de transmissão. Nestes experimentos, os mecanismos de monitoramento foram configurados da seguinte forma: a) mecanismo de *polling* efetuando coletas em intervalos de cem milissegundos e b) sFlow com taxa de amostragem $N = 10$. Ambas as taxas de leitura elevadas, de tal forma a melhorar a precisão das leituras efetuadas.

Por se tratar de experimentos distintos, as escalas dos gráficos foram definidas individualmente, de tal forma a permitir uma melhor avaliação das séries de dados. Devido ao longo tempo de duração, as séries foram plotadas utilizando uma ordem que permitia uma melhor visualização das sobreposições, includindo apenas a série observada no hos-

pedeiro H1 (origem do tráfego) e as séries do mecanismo de *polling* e sFlow observadas no elemento de rede SW3, localizado no centro do cenário avaliado.



(a) UDP gerado com *Iperf*.

(b) TCP utilizando *scp*.

(c) *Streaming* de vídeo utilizando VLC.

(d) Zoom na coleta do *streaming*.

Figura 7. Análise da taxa de transmissão com aplicações legadas.

Nas Figuras 7(a) e 7(b) é possível observar a maior precisão do mecanismo de *polling* frente ao mecanismo de amostragem do sFlow. A Tabela 3 apresenta os dados consolidados referentes às transmissões UDP e TCP, evidenciando no caso do mecanismo de *polling* observações de taxa média similares aos valores de H1 e H2, além de apresentar um valor de desvio padrão inferior ao sFlow.

Tabela 3. Dados consolidados referentes às transmissões UDP e TCP.

Análise	Média e Desvio Padrão (em Mbits/s)			
	$Origem_{H1}$	$Destino_{H2}$	$Polling_{SW1..5}$	$sFlow_{SW1..5}$
Figura 7(a)	1,064 +/- 0,038	1,064 +/- 0,041	1,067 +/- 0,090	1,076 +/- 0,168
Figura 7(b)	1,045 +/- 0,102	1,045 +/- 0,104	1,054 +/- 0,126	1,086 +/- 0,237

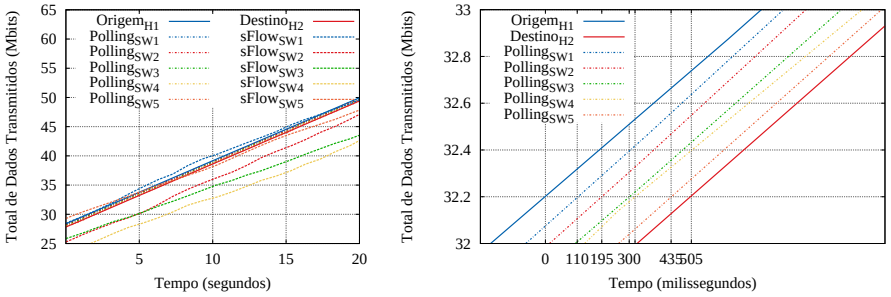
Especificamente para o *streaming* de vídeo apresentado na Figura 7(c), a natureza da aplicação possui variação frequente na taxa de transmissão, não justificando as análises de média e desvio padrão. Desta forma, a Figura 7(d) apresenta um intervalo de aproximadamente três minutos, a contar do inicio do experimento, onde fica caracterizado comportamento similar ao observado nas análises com UDP e TCP, evidenciando a maior precisão no caso do mecanismo de *polling*.

4.4. Análise do Atraso

A Figura 8 apresenta as séries temporais referentes ao total de dados transmitidos, coletadas durante uma transmissão UDP à taxa de 1 Mbps feita pelo gerador de tráfego *Iperf*, em um cenário sem a ocorrência de descarte de pacotes. As medições do mecanismo de

polling ocorreram em intervalos de cem milissegundos e a taxa de amostragem utilizada para o sFlow foi $N = 10$.

Neste experimento, o Mininet foi configurado para aplicar um atraso de cem milissegundos em cada um dos enlaces entre os elementos de rede OpenFlow, ou seja, espera-se que o tráfego experimente em torno de seiscentos milissegundos de atraso fim-a-fim. De maneira geral, o Mininet não conseguiu aplicar o tempo esperado em cada um dos enlaces, realizando um atraso fim-a-fim da ordem de quinhentos milissegundos. Esta configuração foi utilizada para demonstrar a metodologia de cálculo de atraso proposta, uma vez que sem esta configuração o Mininet apresenta atraso próximo a zero.



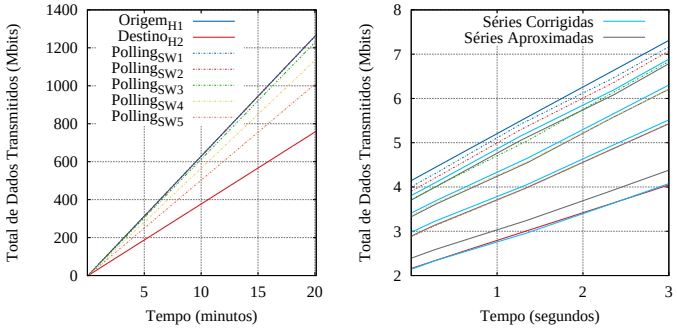
(a) Todas as séries temporais. (b) Zoom nas séries do mecanismo de *polling*.

Figura 8. Análise do atraso sem a ocorrência de descarte de pacotes.

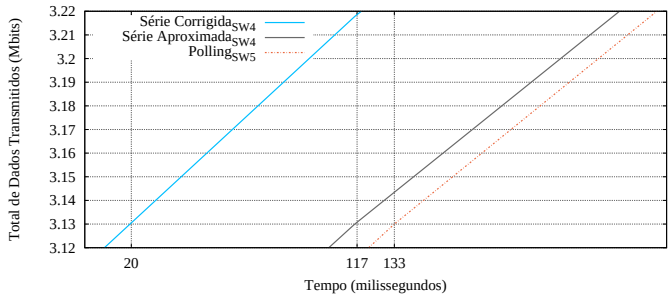
A Figura 8(a) evidencia a inviabilidade do mecanismo de amostragem do sFlow para efetuar a análise de atraso apresentada na Seção 3.3, uma vez que o total de Mbits apresentado nas séries temporais do sFlow são estimativas obtidas através das amostras enviadas ao coletor pelos agentes. Porém, na Figura 8(b) é possível observar o sequenciamento das séries temporais coletadas pelo mecanismo de *polling*, partindo do hospedeiro de origem H1, atravessando sequencialmente os elementos de rede de SW1 a SW5 até atingir o hospedeiro de destino H2. Os valores destacados no eixo X permitem visualizar o atraso entre as séries temporais observadas na altura de 32.2 Mbits transmitidos, sendo 110 ms, 85 ms, 105 ms, 10 ms, 125 ms, 70 ms, respectivamente.

A Figura 9 apresenta as séries temporais coletadas em um cenário onde há descarte de pacotes. Basicamente, foram transmitidos pacotes com MTU de tamanho variado e foram definidos no Mininet enlaces com MTU decrescente entre SW1 (1500 bytes) e SW5 (1496 bytes) para forçar o descarte de pacotes. Foram enviados cinco fluxos UDP com o gerador de tráfego *Iperf*, de tal forma que não haveria nenhum descarte no SW1, uma pequena ocorrência de descarte no SW2, gradualmente aumentando até permitir a vazão de um volume de tráfego no SW5. As medições do mecanismo de *polling* ocorreram em intervalos de cem milissegundos.

A Figura 9(a) apresenta todas as séries temporais durante a transmissão de aproximadamente vinte minutos, deixando evidente o distanciamento entre as séries ocasionado pelos descartes. A Figura 9(b) apresenta as séries corrigidas e aproximadas observadas em um período de três segundos, conforme discutido na Seção 3.3. A Figura 9(c) apresenta ambas as abordagens, detalhando a série temporal coletada do SW5 e as séries corrigida e aproximada referentes ao SW4. É possível observar a significativa perda de precisão do mecanismo aproximado, apresentando um atraso na ordem de 16 milissegundos, sendo



(a) Todas as séries temporais. (b) Séries Corrigidas e Aproximadas.



(c) Comparação entre as séries corrigida e aproximada para o cálculo do atraso.

Figura 9. Análise do atraso quando há a ocorrência de descarte de pacotes.

que a série corrigida apresenta um atraso na ordem de 113 milissegundos.

A coleta referente aos dados descartados foi feita utilizando o contador *Transmit Drops* previsto no OpenFlow apenas para a granularidade em nível de portas. Este contador mantém apenas o número de pacotes descartados. Os resultados da Figura 9 reforçam a sugestão feita na Seção 3.3, para que seja feita a inserção de contadores de total de bytes e total de pacotes descartados para todos os níveis de granularidade em uma futura alteração na especificação do OpenFlow [ONF 2013].

5. Conclusões e Trabalhos Futuros

Este artigo apresentou a plataforma de monitoramento SDNMon, capaz de utilizar diferentes abordagens de monitoramento na obtenção de dados referentes a situação do plano de dados em uma rede OpenFlow. Estes dados são processados e disponibilizados para acesso em outros módulos, tais como a aplicação de monitoramento desenvolvida.

O mecanismo de *polling* proposto neste artigo é capaz de enriquecer a visão da rede mantida pelos controladores OpenFlow, através de monitoramentos precisos e adaptativos à carga experimentada no plano de dados. Este enriquecimento da visão da rede é a base para a plataforma de provimento de qualidade de serviço sendo desenvolvida.

O desenvolvimento deste trabalho permitiu identificar possíveis extensões à especificação OpenFlow, sugerindo a inserção de novos contadores em todos os níveis de granularidade e a possibilidade de especificação de intervalos nas mensagens de *Statistic Request* para permitir uma melhor adaptabilidade por parte do mecanismo proposto.

Finalmente, propomos uma investigação sobre a utilização da abordagem de *push* no envio dos dados dos contadores mantidos pelos elementos de rede do plano de dados OpenFlow na direção do controlador. Ao invés de utilizar amostragens dos cabeçalhos dos pacotes conforme sFlow, o cenário a ser investigado seria o *push* das informações mantidas pelos contadores, evitando as frequentes consultas. A hipótese é que esta abordagem aumentaria a precisão dos monitoramentos e que todas estas medidas contribuiriam para o desenvolvimento de uma ferramenta capaz de prover uma tomografia da rede.

Agradecimento

Os autores gostariam de agradecer CAPES, CNPq e FAPEMIG pelo suporte ao desenvolvimento deste trabalho.

Referências

dos Passos Silva, D., Pontes, A. B., Avelar, E. A. M., and Dias, K. L. (2013). Uma Arquitetura para o Aprovisionamento de QoS Interdomínios em Redes Virtuais baseadas no OpenFlow. *31º Simp. Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Guedes, D., Vieira, L., Vieira, M., Rodrigues, H., and Nunes, R. (2012). Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento de pesquisas em Redes de Computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, 30(4):160–210.

Iperf (2014). `http://iperf.fr`.

Kim, H. and Feamster, N. (2013). Improving network management with software defined networking. *Communications Magazine, IEEE*, 51(2):114–119.

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.

ONF (2013). OpenFlow Switch Specification - Version 1.4.0.

Phaal, P., Panchen, S., and McKee, N. (2001). InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176 (Informational).

Phemius, K. and Bouet, M. (2013). Monitoring latency with OpenFlow. In *Proceedings of the 9th International Conference on Network and Service Management, CNSM 2013, Zurich, Switzerland, October 14-18, 2013*, pages 122–125.

sFlow (2014). `http://sflow.org`.

Suh, J., Kwon, T. T., Dixon, C., Felter, W., and Carter, J. B. (2014). Opensample: A low-latency, sampling-based measurement platform for commodity SDN. In *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*, pages 228–237.

van Adrichem, N. L. M., Doerr, C., and Kuipers, F. A. (2014). OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks. In *2014 IEEE Network Operations and Management Symposium, NOMS 2014, Krakow, Poland, May 5-9, 2014*.

VLC (2014). `http://www.videolan.org/vlc/`.

RafStore: Armazenamento Confiável, Rápido e Geo-Replicado em Provedores de Nuvens

Hylson Netto[1,2], Tulio Alberton Ribeiro[1], 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,tulio.ribeiro,aldelir.luiz}@posgrad.ufsc.br,
lau.lung@ufsc.br, miguel.p.correia@tecnico.ulisboa.pt

Resumo. *Nos últimos tempos, o armazenamento distribuído de dados tem tido grande notoriedade em termos de investigação, em face à crescente oferta de provedores de armazenamento em nuvens. Aplicações que lidam com armazenamento distribuído em nuvens devem considerar aspectos de confiabilidade e disponibilidade. Este trabalho apresenta o RafStore, um protocolo que provê consistência regular, disponibilidade e tolerância a faltas bizantinas. Além disso, o desempenho do RafStore é melhor do que protocolos anteriores e os custos são reduzidos com o uso de menos provedores de dados, para aplicações nas quais o dado a ser armazenado não depende do dado armazenado previamente.*

Abstract. *Lately, distributed data storage has been gaining a lot of attention due the advent of a variety of cloud storage offerings. Applications that deal with distributed storage in clouds have to balance aspects such as reliability and availability. We present RafStore a protocol that provides regular consistency, availability and Byzantine fault tolerance. Additionally, RafStore's performance is better than previous works and costs are reduced by using fewer providers to store data, for applications in which the data to be stored does not depend on the data stored.*

1. Introdução

O paradigma introduzido pelo conceito de Computação em Nuvem (*Cloud Computing*) [NIST 2011] tem impelido no surgimento de uma série de facilidades para a realização de computações através da Internet. Uma das facilidades mais exploradas pelas aplicações na Internet é o armazenamento de dados em nuvem. De um modo geral, aplicações da Internet lidam com clientes que estão dispersos geograficamente, razão pela qual, elas tendem a armazenar dados de maneira distribuída, a fim de manter dados próximos dos clientes no intuito de prover melhor desempenho acerca das operações do cliente. Provedores de armazenamento estão disponíveis em nuvens, trazendo os benefícios de infraestruturas elásticas e custos sob demanda. O número de provedores de nuvens públicas tem aumentado, para armazenamento e para processamento de dados; assim, é possível disponibilizar aplicações e dados na Internet hospedando dados e programas em provedores de nuvens. Provedores passivos, como o Amazon S3 e o Google Cloud Datastore, são ideais para armazenar dados pois possuem custos reduzidos e mecanismos de acesso padronizados, o que torna a mudança de provedor fácil sob a perspectiva do cliente. Estes provedores passivos de armazenamento de dados seguem o modelo de Software como serviço (SaaS).

Os provedores de nuvens são distribuídos geograficamente em *data centers*, de modo que as aplicações precisam escolher um *data center* para armazenar seus dados. É evidente que para se obter um melhor desempenho, os dados devem permanecer próximos às aplicações. Uma estratégia bastante comum empregada para esta finalidade é a replicação geográfica de dados (i.e., geo-replicação), de modo a permitir que as aplicações obtenham os dados do *data center* mais próximo ao qual a aplicação está localizada. Por outro lado, a literatura aponta para o fato de que a geo-replicação introduz alguns problemas; o teorema CAP [Brewer 2012] declara que é impossível alcançar consistência, disponibilidade e particionamento de redes simultaneamente em dados replicados sobre redes de longa distância, como a Internet. Desde o advento deste anúncio – em 2002 – muitos trabalhos têm focado no equilíbrio destas dimensões com o objetivo de satisfazer demandas das aplicações.

Em se tratando de dados, uma característica de particular interesse em sistemas de armazenamento é a relação entre valores atuais e novos valores. No caso, se um dado precisa considerar o valor atual para criar um novo valor, pode-se considerar que novos valores deste dado são *dependentes* de valores anteriores. Esta característica implica inclusive que o dado seja atualizado de uma maneira transacional para evitar leituras sujas. Por exemplo, para um contador compartilhado o valor atual precisa ser lido e incrementado, e então o novo valor pode ser escrito. De outro modo, se um dado pode ser escrito no sistema sem depender do valor anterior, a operação de escrita é semelhante a um *blind-write* [Agrawal and Krishnaswamy 1991], ou uma operação de somente-escrita. Para fins de brevidade, vamos denominar este tipo de dado como dado de *conteúdo independente*. Por exemplo, em um fórum, quando um usuário publica um tópico, outros usuários podem adicionar comentários sobre o tópico. Estes comentários podem estar relacionados ao tópico, ou não apresentar nenhuma relação com o tópico - por exemplo, um usuário oportunista pode enviar um texto que contém propaganda sobre um assunto qualquer. Além disso, para alguns fóruns não há problema se novas mensagens aparecem entre a leitura do usuário e o envio do novo comentário – uma *postagem suja*. A solução proposta no âmbito deste artigo é baseada em aplicações que manipulam dados com *conteúdo independente*.

Sistemas de armazenamento baseados na replicação de dados de *conteúdo independente* buscam obter bom desempenho sem sacrificar a consistência. Um exemplo concreto[1] é o Facebook, que utiliza replicação primário-secundário [Budhiraja et al. 1993] sobre diferentes regiões geográficas. No caso, as réplicas secundárias visam atender requisições de leitura, de modo a reduzir a latência entre países, por exemplo, de 70ms para 2ms; nesse ínterim, requisições de escrita são redirecionadas para uma réplica primária. Como esta arquitetura (*escreve no primário e lê de um secundário próximo*) poderia levar um usuário a não visualizar a sua própria publicação, uma estratégia é usada para garantir que um escritor sempre visualizará suas próprias publicações: depois que um usuário publica algo, todas as requisições relacionadas à publicação do usuário são direcionadas à réplica primária por algum tempo. Este procedimento proporciona um equilíbrio entre desempenho e consistência, mas se a réplica primária falhar enquanto estiver replicando os dados para as réplicas secundárias, estados inconsistentes podem surgir, já que depois do redirecionamento temporário de requisições de leitura o usuário poderá não visualizar o valor que escreveu. Não obstante, sistemas baseados na replicação primário-secundário não são passíveis de tolerar faltas de natureza arbitrária (i.e., bizantinas) [Lamport et al. 1982].

[1] http://www.podc.org/podc2012-report/

Na literatura alguns trabalhos discorrem sobre armazenamento distribuído com semântica confirmável [Martin et al. 2002] e tolerância a faltas, embora nenhum deles verse especificamente sobre dados de *conteúdo independente*. O DepSky [Bessani et al. 2013] armazena dados em múltiplos provedores de nuvens usando quóruns tradicionais [Malkhi and Reiter 1998], sendo que ele contacta $3f + 1$ provedores de dados em cada passo de comunicação – de acordo com o número mínimo de repositórios necessários para tolerar faltas bizantinas em armazenamento [Martin et al. 2002] – e coleta $2f + 1$ respostas do quórum, onde f é o número máximo de faltas toleradas. No SPANStore [Wu et al. 2013], clientes podem escolher o nível de consistência desejado, o que influencia diretamente no número de provedores nos quais o dado será armazenado. À vista disso, para uma consistência forte, o SPANStore contacta $2f + 1$ provedores e aguarda por $f + 1$ respostas corretas. De outro modo, se apenas a consistência eventual é suficiente, $f + 1$ provedores são contactados, e apenas uma primeira resposta é requerida para concluir o passo de comunicação, enquanto os f restantes provedores são atualizados em segundo plano. O MDStore [Cachin et al. 2014] tolera faltas bizantinas e reduz o número de réplicas de dados de $3f+1$ para $2f+1$. E o Hybris [Dobre et al. 2014] é o único sistema tolerante a faltas bizantinas que tenta armazenar dados em apenas $f + 1$ provedores, utilizando um *timeout* para avaliar o sucesso da operação; se o tempo estipulado é excedido, f provedores adicionais são contactados. É digno de nota que todos estes trabalhos, exceto o Hybris, contactam incondicionalmente f provedores a mais que o número mínimo de confirmações requeridas.

Neste ensejo, com vista para uma melhor utilização dos recursos e um melhor desempenho, este artigo apresenta o RafStore, um protocolo confirmável de armazenamento distribuído que provê consistência regular, disponibilidade de dados e tolerância a faltas bizantinas para aplicações que lidam com dados de *conteúdo independente*. No presente trabalho argumenta-se que é possível prover o armazenamento de dados de *conteúdo independente*, pela contatação de apenas $f + 1$ provedores de dados, para armazenar dados nestes $f + 1$ provedores. Pelo uso da técnica de antecipação de pedidos, o desempenho do RafStore é superior à abordagem tradicional de quóruns, pois o avanço entre fases do protocolo é acelerado. Além disso, os custos são reduzidos pois apenas $f + 1$ provedores de dados são requeridos em cenários previsíveis e de operação correta.

O restante do trabalho está organizado da seguinte forma: a Seção 2 aborda os trabalhos relacionados à proposta; na Seção 3 é definido o modelo de sistema; a Seção 4 detalha a especificação do protocolo proposto; na Seção 5 é apresentada a validação do RafStore; e, a Seção 6 apresenta discussões sobre os resultados. Por fim, a Seção 7 conclui o artigo.

2. Trabalhos relacionados

Na literatura, alguns trabalhos discorrem sobre armazenamento distribuído, tolerância a faltas e níveis de consistência [Bessani et al. 2013, Wu et al. 2013, Cachin et al. 2014, Wang et al. 2012, Cho and Aguilera 2012, Dobre et al. 2014]. O Depsky [Bessani et al. 2013] provê armazenamento distribuído em múltiplas nuvens tolerando faltas bizantinas. No DepSky, a consistência dos dados é garantida a partir de sistema de quóruns [Malkhi and Reiter 1998] e *erasure codes* [Rabin 1989] são usados para reduzir a quantidade de dados armazenados em cada provedor. Apenas um subconjunto de provedores efetivamente armazenam o dado, porém todos os provedores são contactados para realizar o armazenamento, o que resulta em um consumo extra de recursos de rede. Este comportamento segue um limite inferior definido para sistemas de armazenamento

bizantino [Martin et al. 2002], em que $3f + 1$ réplicas são necessárias para garantir que o sistema tolere até f faltas bizantinas. Como exemplo, se for necessário tolerar uma falta, quatro réplicas serão necessárias para compor o sistema.

Uma abordagem mais flexível é realizada pelo SPANStore [Wu et al. 2013], que armazena dados em provedores de nuvens, de modo que cada aplicação pode escolher o nível de consistência desejado. Se uma consistência forte (p. ex.: confirmável) for necessária, o SPANStore contacta $2f+1$ provedores e um quórum de $f+1$ respostas corretas é suficiente para assegurar que o dado está corretamente armazenado. De outro modo, se uma consistência mais fraca for suficiente para a aplicação (p. ex.: consistência eventual), um quórum de $f+1$ provedores é contactado para armazenar os dados, e apenas a primeira resposta correta é suficiente para que a aplicação conclua a operação de escrita; os f provedores restantes serão atualizados de maneira assíncrona, sem confirmações para o cliente de que o dado está corretamente armazenado. Além disso, o SPANStore mantém uma tabela de latências entre os provedores, atualizada a cada hora, com a finalidade de ajudar na recomendação ao usuário sobre em quais provedores o mesmo deve armazenar, a fim de cumprir metas de latência máxima. Note que o SPANStore protege o dado contra faltas de parada, e seu objetivo principal é reduzir custos do dado armazenado.

O protocolo MDStore [Cachin et al. 2014] permite obter uma redução de custos, a partir do armazenamento dos metadados em $3f + 1$ provedores e dos dados em apenas $2f + 1$ provedores, ao invés de $3f + 1$. Embora reduza os custos, aspectos de desempenho não são abordados no MDStore, e a quantidade de $2f + 1$ réplicas de dados é anunciada como um limite inferior para sistemas que armazenam dados e toleram faltas bizantinas. O MDStore usa um serviço de metadados para oferecer armazenamento com semântica linearizável [Herlihy and Wing 1990], enquanto aplicações de *conteúdo independente* geralmente são bem atendidas com consistência regular [Lamport 1978], que é mais fraca do que a semântica linear, mas forte o suficiente para aplicações como redes sociais.

Outro protocolo particularmente interessante é o Gnothi [Wang et al. 2012], o qual replica metadados em todos os provedores enquanto dados são replicados em um subconjunto de provedores preferidos. Quando réplicas tornam-se indisponíveis, réplicas adicionais são contactadas para salvar dados, garantindo que em todas as solicitações ao menos $f + 1$ provedores de dados confirmem a operação de escrita. Um aspecto que é digno de nota é que o Gnothi tolera apenas faltas por parada, mas provê armazenamento linearizável.

O Vivace [Cho and Aguilera 2012] replica dados em provedores remotos e provê consistência forte. Durante congestionamentos de rede, metadados são tornados os menores possíveis e trafegam de maneira priorizada na rede, para permitir a continuidade da operação. O Vivace lida com objetos simples de leitura e escrita, e objetos de leitura, modificação e escrita. Para objetos de leitura e escrita, a linearização é implementada com *write-backs*, que são responsáveis por propagar o último *timestamp* para uma maioria de réplicas. Este passo de comunicação pode ser eliminado com uma otimização denominada paralelização de pedidos: se uma maioria de réplicas não contiver o último *timestamp*, a etapa de *write-back* é disparada de maneira pró-ativa paralelamente à etapa de leitura de dados. Outra otimização no Vivace sugere que o cliente mantenha um histórico de latências dos provedores remotos, a fim de contactar os provedores que possuam menores latências.

E finalmente, o Hybris [Dobre et al. 2014] consiste em um sistema de armazenamento chave-valor que armazena metadados em provedores de nuvens privadas ativas con-

fiáveis e distribui os dados em provedores de nuvens públicas passivas que são até certo ponto (i.e., o limite f de faltas) confiáveis. Os dados são escritos em $f + 1$ provedores no caso normal – sob o atendimento de um *timeout*; e no caso de esgotamento do tempo definido, um total de $2f + 1$ provedores são contactados para efetivar a escrita. Ele tolera faltas de parada nos provedores de metadados e faltas bizantinas nos provedores de dados. Hybris suporta múltiplos escritores e leitores, e é linearizável. A comunicação entre clientes e o serviço de metadados é parcialmente síncrona [Dwork et al. 1988]; entre clientes e nuvens públicas é assíncrona. O serviço de metadados é implementado por máquinas de estado replicadas; provedores de nuvens são eventualmente consistentes e os dados são versionados. Para garantir uma forte consistência dos dados, Hybris permite que clientes inscrevam-se no serviço de metadados para receberem notificações em caso de atualizações de metadados. Isto é útil no caso em que leitores estão lendo dados concorrentemente com escritas.

Uma breve comparação acerca dos trabalhos relacionados demonstra que o MDStore alcança redução de custos, o SPANStore provê uma maneira de saber quais provedores são os mais rápidos e o DepSky obtém melhor desempenho a despeito de um protocolo com maior custo. Hybris tenta reduzir custos utilizando menos provedores de dados, mas torna necessária a utilização de um *timeout*. Neste sentido, à luz dos trabalhos relacionados o protocolo proposto neste trabalho visa: (*i*) reduzir o número de provedores necessários para armazenar os dados, de modo que, para metadados, $3f + 1$ provedores ainda serão necessários; (*ii*) tolerar faltas bizantinas ao tempo que oferece semântica regular; (*iii*) prover maior desempenho sem a utilização de *timeouts* por meio de uma paralelização parcial de requisições. A proposta deste artigo considera latência dos provedores uma informação significativa que o cliente pode considerar para obter melhor desempenho no protocolo de armazenamento. Por fim, o protocolo proposto visa preencher a lacuna de obter simultaneamente, a redução de custos e melhoria de desempenho, bem como pela evocação de réplicas adicionais sob demanda, similar ao que ocorre no Gnothi.

3. Modelo de Sistema

Para a especificação do protocolo proposto, considera-se um ambiente de sistema distribuído assíncrono, em que clientes e provedores de armazenamento estão conectados por uma rede, onde os canais de comunicação são confiáveis (p. ex.: as mensagens enviadas finarão por serem recebidas). Clientes podem ler e escrever dados nos provedores de armazenamento via operações simples como *get* e *put*, respectivamente. Um esquema chave-valor referencia os dados, e na maioria dos casos espera-se que haja apenas um escritor por vez (SWMR – *single writer, multiple readers*) [Lamport 1986]; entretanto, um esquema de baixa contenção [Bessani et al. 2013] pode ser empregado para tolerar múltiplos escritores em provedores de nuvem passivos. Os provedores de armazenamento são entidades passivas, de maneira que não podem executar código ou comunicar-se entre si.

Em relação aos dados manipulados no âmbito do protocolo, estes são considerados como dados de *conteúdo independente*, isto é, para todos os valores escritos, não existe relação de processamento com valores anteriores do dado. Isto significa que uma operação de escrita pode ser executada sem uma leitura prévia do dado para compor o novo valor, de forma semelhante a uma escrita direta, ou *blind-write* [Agrawal and Krishnaswamy 1991]; isto simplifica a especificação e aumenta o desempenho do protocolo. No que concerne às faltas, adota-se um modelo híbrido de falhas, de modo que os clientes podem falhar de maneira arbitrária e exibir comportamento bizantino enquanto estiverem lendo dados –

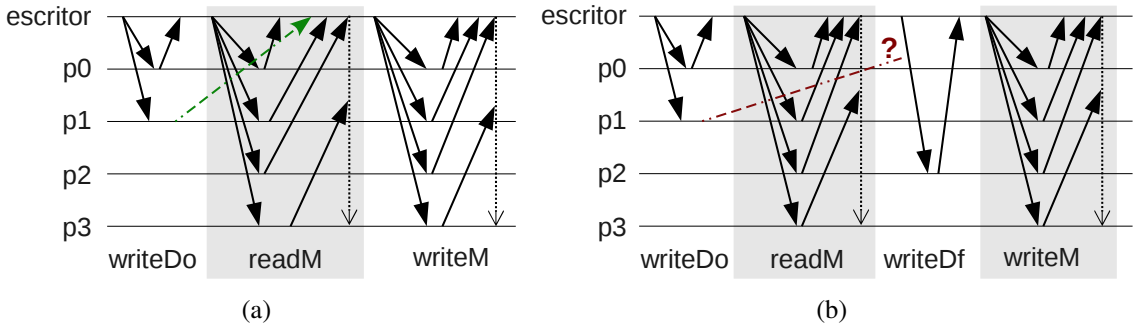


Figura 1. Protocolo RafStore: operação de escrita em (a) execução normal e (b) execução com faltas ou em condições desfavoráveis de rede; configuração com f=1 e n=4.

visto que estes clientes não estarão modificando o estado do sistema; entretanto clientes podem falhar somente por parada quando estiverem escrevendo dados: clientes maliciosos poderiam, por exemplo, escrever valores diferentes, em provedores diferentes, associados a uma mesma versão de dado, ou mesmo exaurir o espaço de versionamento utilizando valores de versão muito grandes, e não é objetivo deste trabalho tratar destas questões[2]. De outro modo, os provedores de armazenamento podem ser corretos ou apresentar comportamento bizantino. O número de provedores responsáveis por armazenar metadados é igual a $3f+1$, sendo que não mais de $f \leq \lceil \frac{n-1}{3} \rceil$ provedores podem falhar simultaneamente, durante uma janela de vulnerabilidade do sistema.

4. O Protocolo RafStore

O RafStore consiste em um protocolo cuja especificação visa prover não apenas um ambiente confiável, mas também otimizado para o armazenamento de dados de *conteúdo independente*. Neste sentido, em condições normais de execução o armazenamento ocorre em $f+1$ provedores de armazenamento, ou, de outro modo, em $2f+1$ provedores de armazenamento quando ocorrem faltas ou quando a rede não apresenta condições favoráveis, conforme explicação detalhada adiante[3]. O RafStore tolera faltas bizantinas a partir do uso de $3f+1$ provedores para armazenar metadados. Para um melhor entendimento, a Figura 1 ilustra o padrão de comunicação adotado para operações de escrita no RafStore.

O funcionamento do protocolo se dá da seguinte maneira, o cliente inicia a escrita de dados (fase *writeDo* da figura 1(a)) por meio de uma solicitação a $f+1$ provedores de dados, para que armazenem o dado. Quando a primeira resposta de um provedor de dados chega ao cliente, a próxima fase (*readM*) é iniciada de maneira especulativa, em que os metadados são buscados junto a provedores que armazenam os metadados – este aspecto consiste na técnica de antecipação de pedidos mencionada na Seção 1. Quando são obtidas respostas suficientes (i.e., $2f+1$) da fase *readM* pelo escritor, uma verificação crucial ocorre: se todas as $f+1$ respostas concernentes à fase *writeDo* foram obtidas dos provedores de dados, os novos metadados podem ser escritos e assim a fase *writeM* é iniciada, conforme ilustrado na figura 1(a). Se ao término da fase *readM* não foi obtido um número suficiente de respostas – conforme ilustra a Figura 1(b) – uma fase adicional nomeada *writeDf* é executada, onde f provedores adicionais de armazenamento de dados são contactados para assegurar que, pelo menos $f+1$ provedores de armazenamento irão armazenar o dado. Note pela Figura 1(b), que tanto a última resposta da fase *writeDo* quanto a resposta da fase *writeDf* são válidas

[2]Existem maneiras de tratar estas questões, como fases adicionais de *write-back* durante operações de leitura para que clientes corretos façam a reparação dos dados [Liskov and Rodrigues 2006].

[3]O texto próximo à Figura 3 detalha estas condições de rede.

Algoritmo 1 Operação de escrita no RafStore

```
1:  procedure WRITE(key, value)
2:      uId := UUID() + "_" + hash(value)                    ▷ identificador único
3:      global c := 0                  ▷ contador de respostas válidas; ver algoritmo 3
4:      global p := ∅                  ▷ conjunto de respostas válidas; ver algoritmo 3
5:      wQuorum(uId, value, 0, f)      ▷ solicita escrita em f + 1 provedores de dados
6:      wait until c ≥ 1
7:      m := readMDData(key, n − f)                                  ▷ ler metadados
8:      if c < (f + 1) then
9:          wQuorum(key, value, f + 1, 2 ∗ f)
10:     wait until c ≥ f + 1
11:     parallel for 0 ≤ i ≤ 2 ∗ f do provider_i.cancelPending()
12:     newV := max(m[i].v : 0 ≤ i ≤ 2 ∗ f) + 1
13:     newMDData := newV + "; " + uId + "; " + p
14:     newMDData := m + newMDData
15:     mt :=⊥                                   ▷ inicializa conjunto de respostas
16:     parallel for 0 ≤ i ≤ n − 1 do
17:         mt[i] := provider_i.put("metadata"+key,newMDData)
18:     wait until |{∀i : mt[i] = "ok"}| ≥ n − f
19:     parallel for 0 ≤ i ≤ n − 1 do provider_i.cancelPending()
```

para completar o quórum de respostas esperadas na fase *writeDf*. Com a estratégia de antecipação de pedidos, é necessário ligar o dado e o metadado visto que o nome do dado armazenado não está relacionado à chave. Para atender a esta necessidade, um identificador único pode ser representado pela composição de um identificador UUID [Leach et al. 2005] aleatório e um *hash* do dado ($MD5$). Neste sentido, uma convenção para representação de metadado no RafStore é através de uma tupla (v,uId,($r_0, r_1, ..., r_f$)), em que v significa a versão do dado, uId é o identificador do dado e r_i é o i-ésimo provedor de dados. Um único arquivo de metadados contém os metadados de todas as versões do dado.

O Algoritmo 1 apresenta detalhadamente a operação de escrita do RafStore. O cliente solicita a escrita do dado com o comando *write(key, value)* (linha 1). O dado é associado a um identificador único (linha 2) e enviado a todos os provedores de dados (linha 5). Quando a primeira resposta dos provedores de dado chega ao cliente (linha 6), o cliente solicita aos provedores de metadados a leitura dos metadados (linha 7). Quando um quórum de respostas sobre a leitura de metadados é obtido, é verificado se todas as escritas de dados foram concluídas (linha 8); se esta verificação detectar que existem escritas de dados ainda não confirmadas, mais provedores de dados são contactados para armazenar o dado (linha 9). De qualquer maneira, a operação de escrita poderá avançar apenas quando pelo menos $f + 1$ provedores de dados confirmarem que o dado foi salvo (linha 10). Para minimizar possíveis desperdícios, eventuais escritas pendentes adicionais são canceladas (linha 11); esta situação ocorre quando provedores de dados adicionais são contactados (linha 9). No que segue, uma nova versão do dado é calculada (linha 12) e o metadado anterior é anexado ao novo metadado (linhas 13 e 14). Por fim, todos os provedores de metadados são contactados para armazenar o metadado (linha 17), de modo que um quórum de $2f + 1$ respostas é suficiente (linha 18) para concluir esta etapa; solicitações de escrita de metadado remanescentes são canceladas para concluir a operação de escrita (linha 19).

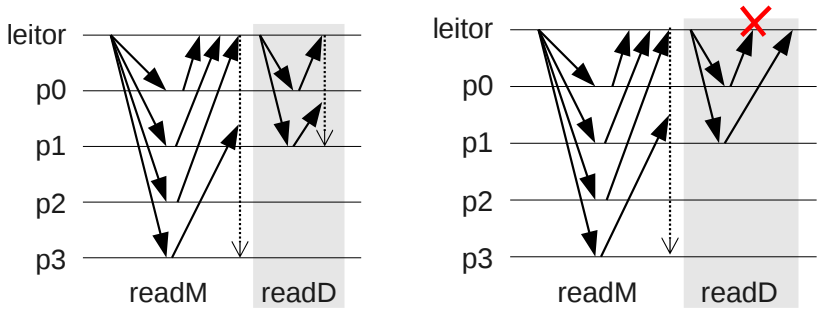


Figura 2. Protocolo RafStore: operação de leitura em (a) execução normal e (b) execução com faltas; configuração com $f = 1$ e $n = 4$.

Algoritmo 2 Operação de leitura no RafStore

```
1: function READ(key)
2:     m := readMData(key, n − f)
3:     parallel for i in m.providersId do
4:         data[i] := provider_i.get(m.getUniqueId())
5:     wait until data[i] is valid
6:     parallel for i in m.providersId do provider_i.cancelPending()
7:     return data[i]
```

No que segue, a operação de leitura do protocolo é apresentada na Figura 2, cuja especificação ocorre no Algoritmo 2. O cliente solicita o metadado aos provedores de metadados, a partir do fornecimento da chave (linha 2); quando um número suficiente de respostas chega ao cliente ($2f + 1$), o dado é solicitado aos provedores de dados que contém aquele dado armazenado (linha 4). Quando um dado válido é recebido de um provedor de dados (linha 5), as solicitações remanescentes de leitura de dados são canceladas (linha 6) e o valor obtido dos provedores pode ser entregue ao cliente (linha 7). Procedimentos auxiliares para o funcionamento do protocolo são especificados no Algoritmo 3. O procedimento *wQuorum* (linha 1) contacta um conjunto de provedores p_{ini} a p_{end} para armazenar um valor sob a denominação de uma chave. Na medida em que os provedores confirmam a operação de gravação (linha 4), variáveis globais (linhas 5 e 6) são atualizadas para refletirem o tamanho do quórum de respostas. A função *readMData* (linha 7) solicita aos provedores de metadados o metadado da chave desejada (linha 10). Quando a quantidade necessária de respostas corretas é alcançada (linha 13), solicitações de leitura pendentes são canceladas (linha 14) e o metadado é retornado para o cliente (linha 15).

Um aspecto relevante em se tratando do protocolo proposto, decorre do fato de que é natural supor que a operação de escrita do RafStore alcançará melhor desempenho do que em outras abordagens, visto que a quantidade de provedores de dados contactados é menor. Entretanto, o melhor desempenho estimado é condicionado à escolha de quais provedores atuarão como provedores de dados. Para tanto, considere os provedores (p_0, ..., p_3) e seus respectivos RTT's (*round-trip time*) apresentados na Figura 3. Em ambos os cenários existe uma réplica de dados com alto valor de RTT, beneficiando a tolerância a faltas catastróficas por meio da geo-replicação de longa distância. A Figura 3(a) mostra um cenário onde o provedor de dados mais lento tem RTT igual a 70ms; neste caso, quando a fase *readM* (representada pela sombra cinza) terminar, a resposta do segundo provedor de dados já terá

Algoritmo 3 Operações auxiliares do protocolo

```
1:  procedure WQUORUM(key, value, ini, end)
2:      parallel for ini ≤ i ≤ end do
3:          ok[i] := provider_i.put(key, value)
4:          if ok[i] = "ok" then
5:              c := c + 1                          ▷ incrementa contador de respostas válidas
6:              p := p ∪ {i}                        ▷ adiciona resposta ao conjunto de respostas válidas

7:  function READMDATA(key, threshold)
8:      m :=⊥
9:      parallel for 0 ≤ i ≤ n − 1 do
10:         t_i := provider_i.get("metadata"+key)
11:         if verify(t_i) then
12:             m[i] := t_i
13:     wait until |{∀i : m[i] ≠⊥}| ≥ threshold
14:     parallel for 0 ≤ i ≤ n − 1 do provider_i.cancel_pending()
15:     return m
```

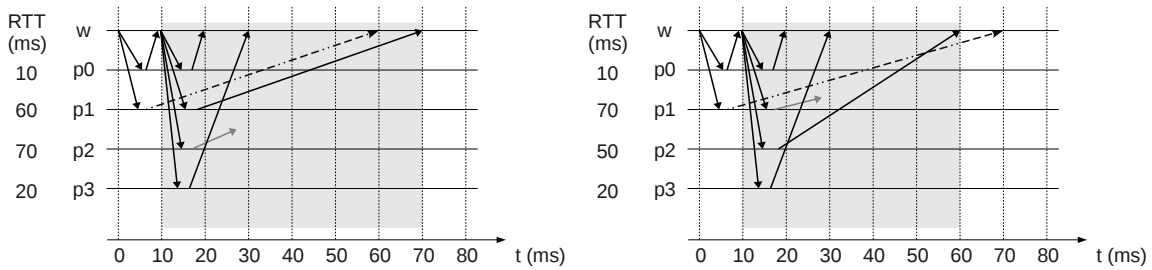


Figura 3. Exemplos de RTT: (a) execução normal e (b) condições desfavoráveis de rede, ou cenário de faltas; configuração com $f = 1$ e n=4.

chegado ao cliente, permitindo que o cliente avance para a fase *writeM* (não desenhada na figura) e conclua a operação de escrita. Entretanto, se o RTT do segundo provedor de dados for o maior entre o conjunto de provedores, a resposta do segundo provedor de dados não terá chegado ao cliente quando a fase *readM* terminar – como mostrado na Figura 3(b) – e outro provedor de dados precisará ser contactado, conforme representado na Figura 1(b).

Análises no comportamento do RafStore revelam que existe uma condição determinante que separa os cenários mostrados nas Figuras 3(a) e 3(b). Esta condição depende de três valores: a, definido na equação 1, o qual representa o provedor de dados mais lento; b, definido na equação 2, que refere-se ao provedor de dados mais rápido; e c, definido na equação 3, que denota o provedor de metadados mais lento no subconjunto dos $2f + 1$ provedores de metadados mais rápidos. RTT é o *round-trip time*, dp_i é o i-ésimo provedor de dados, mdp_i é o i-ésimo provedor de metadados e $subset(set, k)$ retorna os primeiros k elementos de set. O termo $ordered$ significa que o item anexo está ordenado. A condição que garante um melhor desempenho no protocolo RafStore é que a precisa ser menor ou igual ao valor $(b + c)$. Esta relação está expressa textualmente na Condição 1.

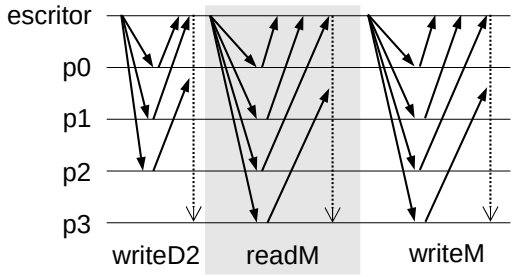


Figura 4. Modo de operação tradicional no RafStore: útil quando não há informações sobre latências dos provedores.

$$a = max(RTT_{dp_i}), 0 < i < n_{dp} - 1 \quad (1)$$

$$b = min(RTT_{dp_i}), 0 < i < n_{dp} - 1 \quad (2)$$

$$c = max(subset(orderedRTT_{mdp_j}, 2f + 1)), 0 < j < n_{mdp} - 1 \quad (3)$$

Condição 1: *a escolha dos provedores deve obedecer a regra* $a < (b + c)$. *Em outras palavras, o provedor de dados mais lento precisa ser mais lento ou tão rápido quanto a soma do provedor de dados mais rápido e o provedor de metadados mais lento no subgrupo dos* $2f + 1$ *provedores de metadados mais rápidos.*

Quando não é possível atender a Condição 1, pode-se proceder de um modo mais tradicional. A Figura 4 ilustra uma operação com quóruns tradicionais, onde um quórum de $2f+1$ provedores de dados é contactado para armazenar dados (fase *writeD2*). Quando $f+1$ respostas dos provedores de metadados chegam ao cliente, solicitações remanescentes são canceladas e as próximas duas fases (*readM* e *writeM*) são iniciadas para realizar a leitura e a escrita de metadados. O algoritmo deste modo de operação tradicional foi omitido por questões de espaço, mas é possível derivá-lo a partir dos algoritmos já apresentados.

O conhecimento prévio do RTT dos provedores é útil para orientar as escolhas de quais provedores serão responsáveis pelo armazenamento de dados e quais armazenarão apenas metadados. Algumas recomendações sugerem uma arquitetura que ofereça as condições ideais para o RafStore; esta arquitetura é apresentada pela Figura 5:

- f provedores devem ser do tipo I: estes provedores devem estar *próximos* ao cliente, para obter melhor **desempenho**. Estes provedores são replicados em quantidade igual ao número de faltas toleradas no sistema, e podem ser nuvens menos custosas ou nuvens privadas. Dados e metadados serão armazenados nestes provedores.
- ao menos um provedor deve ser do tipo II: este provedor deverá ser robusto e hospedar uma cópia de dados; deve estar *distante* para garantir a **sobrevivência** do dado.
- f provedores devem ser do tipo III: estes provedores devem estar tão distantes quanto o provedor de dados mais distante, a fim de manter válida a relação $a < (b + c)$. Além disso, estes provedores são responsáveis principalmente pelo armazenamento de metadados, mas podem armazenar dados em caso de faltas.
- não há recomendações especiais para os f provedores remanescentes, denominados do tipo IV. Estes provedores armazenam apenas metadados.

5. Experimento

De acordo com as características apresentadas na Seção 4, o RafStore deve alcançar melhor desempenho do que protocolos baseados em quóruns tradicionais quando as condições de

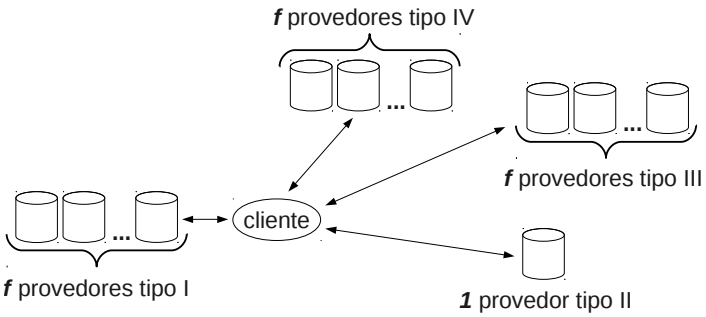


Figura 5. Clientes e provedores: arquitetura sugerida para o uso do protocolo RafStore.

rede (latências) forem favoráveis. Esta afirmação remete à primeira hipótese:

Hipótese 1: *se as latências dos provedores de dados obedecerem à Condição 1, o protocolo RafStore alcançará desempenho melhor que os protocolos de quóruns tradicionais.*

Outro ponto a ser investigado é se o tamanho do dado interfere na diferença do desempenho entre os protocolos considerados. Esta ponderação é intuitivamente verdadeira, de modo que a literatura evidencia [Padilha and Pedone 2011] que existe um valor limite até o qual operações de escrita e leitura tem desempenhos similares, e a partir do qual a diferença de desempenho será maior. Esta asserção remete à segunda hipótese:

Hipótese 2: *na medida em que o tamanho do dado aumenta, a diferença de desempenho entre os protocolos comparados também aumenta.*

Um experimento foi conduzido para avaliar as hipóteses apresentadas. O tempo de resposta percebido pelo cliente foi a informação coletada. Os testes foram executados em um sistema distribuído, com quatro computadores representando os provedores e um quinto computador representando o cliente. Neste cenário, de acordo com a regra $3f + 1$, uma falta bizantina é tolerada ($f = 1$). Os provedores foram executados em computadores com a configuração Intel i7 3.5Ghz, QuadCore, cache L3 8MB, 12GB RAM, 1TB HD. O cliente foi executado sobre um computador com configuração Intel i7 3.07GHz, QuadCore, cache L3 8MB, 16GB RAM, 2TB HD. Todas as máquinas executaram o sistema operacional Debian 7.4 Wheeze, 64 bits, kernel 3.2.54-2, com JVM Oracle JDK 1.7. Clientes e provedores foram interconectados por uma rede Ethernet 10/100 MBits.

Para verificar a Hipótese 1, as latências dos provedores serão definidas com valores de acordo com a Condição 1. Os valores escolhidos foram (0, 161, 308, 181) para os provedores p_0, ..., p_3, respectivamente. Estes valores foram obtidos de um trabalho relacionado [Terry et al. 2013] e representam latências para um cliente na China e provedores em *data centers* no Oeste dos Estados Unidos, Inglaterra e Índia, respectivamente.

Note que o RafStore pode operar em dois modos: o caso quando as latências obedecem a Condição 1 e apenas $f + 1$ provedores são contactados para armazenar dados (uso da técnica de antecipação de pedidos), ou um caso mais tradicional, onde $2f + 1$ provedores de dados são contactados. Estas duas abordagens buscam obter $f + 1$ confirmações de que provedores de dados realizaram o armazenamento. Neste sentido, o RafStore será comparado nestes dois modos de operação, em relação aos protocolos especificados no DepSky [Bessani et al. 2013]. O tamanho do dado será variado de 1KB até 1MB, por um fator multiplicativo de 4, com a finalidade de verificar a Hipótese 2. Estes valores cobrem desde o tamanho de uma frase textual grande até o tamanho típico de uma fotografia. Para simular o RTT, os computadores provedores foram configurados com emulador de rede

WAN[4]. A unidade de tempo considerada foi *ms*, provedores de metadados são definidos como p_0, ..., p_3 e provedores de dados são definidos como p_0 e p_1. A ordem de execução dos pedidos, para os diferentes tamanhos e protocolos, foi sorteada para prover aleatoriedade ao experimento. Vinte replicações do experimento foram realizadas; os dados coletados ajustam-se bem a uma distribuição normal e estão disponíveis em hylson.com/rafstore.

É digno de nota que o RafStore foi implementado em Java. Para uma comparação mais justa, os protocolos do DepSky foram reimplementados, sem otimizações, sobre a mesma base de código do RafStore. O DepSky-A realiza replicação integral de dados, enquanto o DepSky-CA requer o uso de *erasure codes* [Rabin 1989], que foram utilizados da biblioteca JEC, disponível no código do DepSky[5]. Aspectos de privacidade não foram implementados, visto que o objetivo principal do experimento é comparar o desempenho observando o número de provedores de dados e o tamanho do dado armazenado nos provedores de dados. No DepSky-CA, o uso de *erasure codes* reduz o tamanho dos dados armazenados em cada provedor de dados. O RafStore está disponível em hylson.com/rafstore.

6. Avaliação

A realização do experimento descrito na Seção 5 nos permitiu verificar alguns aspectos em relação ao RafStore. A Figura 6 apresenta as latências de operações de escrita para os diferentes protocolos comparados: DepSky-A, DepSky-CA, RafStore usando a abordagem de pedidos antecipados e RafStore usando a abordagem tradicional. Nota-se que apenas o RafStore com a abordagem de pedidos antecipados obtém melhor desempenho do que os outros métodos, para todos os tamanhos de dados considerados. Com relação à Hipótese 1, os resultados não permitem concluir que o RafStore supera, em desempenho, os protocolos de quóruns tradicionais, mesmo quando as latências dos provedores atendem à Condição 1. Entretanto, é possível afirmar que o RafStore que utiliza pedidos antecipados apresenta melhores resultados que os demais protocolos, para todos os tamanhos de dados considerados.

Sobre a relação entre o tamanho do dado e a variação de desempenho, é possível observar, por exemplo, que a diferença de desempenho entre os protocolos DepSky-A e DepSky-CA aumenta conforme o dado aumenta; porém, essa relação não ocorre continuamente entre os protocolos DepSky-CA e RafStore antecipado, pois a diferença aumenta quando o dado cresce de 64KB para 256KB, mas diminui na transição de 256KB para 1MB. Assim, considerando a Hipótese 2, não é possível afirmar que a diferença de desempenho entre os protocolos comparados aumenta de acordo com o aumento do tamanho do dado. Por fim, destaca-se que o RafStore com o uso de pedidos antecipados alcança uma latência em torno de 30% menor que o protocolo DepSky-CA, para dados maiores que 64KB; para dados com 1MB de tamanho, essa diferença diminui para cerca de 21%.

Um comportamento interessante pode ser percebido na Figura 6: o desempenho do RafStore no modo de operação tradicional comparado com o protocolo DepSky-CA. Para dados de tamanho 256KB, o RafStore possui melhor desempenho do que o DepSky-CA. Para dados de tamanho 1024KB, o DepSky-CA possui melhor desempenho do que o RafStore. Essa inversão pode ser explicada pela fragmentação de dados realizada pelo DepSky-CA: visto que fragmentos são salvos mais rapidamente do que dados integrais (pois são menores), é evidente que o DepSky-CA obtém um melhor desempenho em relação ao

[4]http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[5]http://code.google.com/p/depsky/

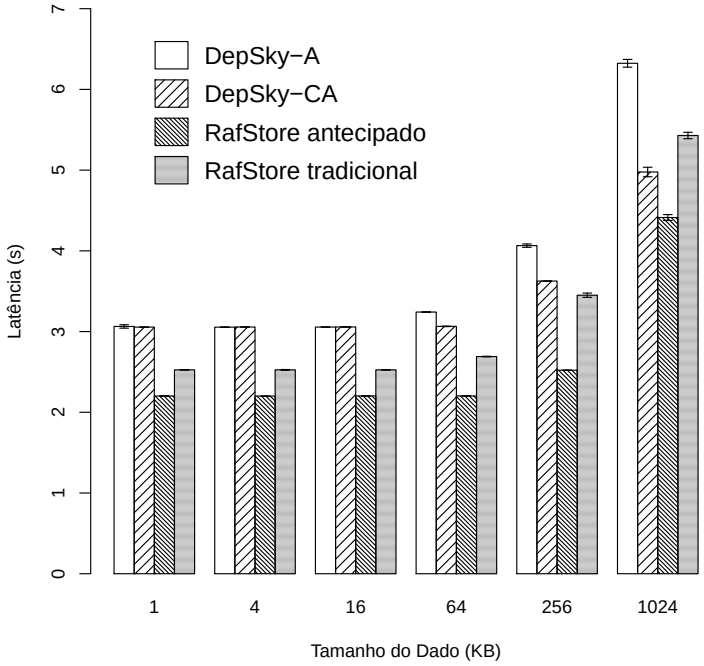


Figura 6. Latências para os métodos comparados, com variação no tamanho do dado.

RafStore, quando o tamanho dos dados aumenta. Entretanto, existem outras questões sobre o tamanho de fragmentos em relação ao tamanho integral de dados. Um micro-experimento foi conduzido utilizando o código do DepSky para comparar o tamanho dos fragmentos gerados pelo DepSky-CA, na medida em que o tamanho do dado aumenta. De acordo com os dados da Tabela 1 existe um ponto a partir do qual o tamanho do dado armazenado como fragmento (DepSky-CA) é menor do que o tamanho integral do dado. Uma discussão mais aprofundada sobre tais aspectos será desenvolvida em trabalhos futuros.

Tabela 1. Tamanho do fragmento, em bytes, para o DepSky-(A) e DepSky-(CA).

DepSky-(A)	1	512	1024	1536	2048	2560	3072	3584
DepSky-(CA)	685	941	1197	1453	1709	1965	2221	2477

7. Conclusão

Este artigo apresentou o RafStore, um protocolo de armazenamento distribuído que possui desempenho melhor que trabalhos anteriores – até 30% em alguns casos – e consome menos recursos – dispensa o uso de um provedor de nuvens para armazenar dados – se algumas condições de rede, como latência entre o cliente e os provedores, puderem ser observadas, e a aplicação fizer uso de dados de conteúdo independente. Não obstante, o protocolo do RafStore foi especificado para prover não somente geo-replicação, mas também para tolerar faltas bizantinas. Os clientes precisam escolher quais provedores e *data centers* desejam usar para armazenar dados em nuvens e o RafStore pode orientar esta escolha se desempenho e economia de recursos forem prioridades do cliente.

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.

Referências

Agrawal, D. and Krishnaswamy, V. (1991). Using multiversion data for non-interfering execution of write-only transactions. *SIGMOD Record*, 20(2):98–107.

Bessani, A., Correia, M., Quaresma, B., André, F., and Sousa, P. (2013). DepSky: Dependable and secure storage in a cloud-of-clouds. *ACM Transactions on Storage*, 9(4):12:1–12:33.

Brewer, E. (2012). CAP twelve years later: How the "rules"have changed. *Computer*, 45(2):23–29.

Budhiraja, N., Marzullo, K., Schneider, F. B., and Toueg, S. (1993). The primary-backup approach. In Mullender, S., editor, *Distributed systems*, pages 199–216. 2 edition.

Cachin, C., Dobre, D., and Vukoli, M. (2014). Separating data and control: Asynchronous BFT storage with 2t+1 data replicas. In *Proceedings of the 16th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 1–17.

Cho, B. and Aguilera, M. K. (2012). Surviving congestion in geo-distributed storage systems. In *Proceedings of the USENIX Annual Technical Conference*, pages 439–451.

Dobre, D., Viotti, P., and Vukolić, M. (2014). Hybris: Robust hybrid cloud storage. In *Proceedings of the 5th annual ACM Symposium on Cloud Computing*, pages 1–14.

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

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

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565.

Lamport, L. (1986). On interprocess communication. *Distributed Computing*, 1(2):77–101.

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

Leach, P., Mealling, M., and Salz, R. (2005). A universally unique identifier (UUID) URN namespace (RFC 4122). IETF Request For Comments.

Liskov, B. and Rodrigues, R. (2006). Tolerating byzantine faulty clients in a quorum system. In *the 26th IEEE International Conference on Distributed Computing Systems*, pages 34–43.

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

Martin, J.-P., Alvisi, L., and Dahlin, M. (2002). Minimal byzantine storage. In *Proceedings of the 16th International Conference on Distributed Computing*, pages 311–325.

NIST (2011). National institute of standards and technology definition of cloud computing. Disponível em `http://www.nist.gov/itl/cloud/`.

Padilha, R. and Pedone, F. (2011). Belisarius: BFT storage with confidentiality. In *Proceedings of the 10th IEEE International Symposium on Network Computing and Applications*, pages 9–16.

Rabin, M. O. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348.

Terry, D. B., Prabhakaran, V., Kotla, R., Balakrishnan, M., Aguilera, M. K., and Abu-Libdeh, H. (2013). Consistency-based service level agreements for cloud storage. In *Proceedings of the 24th Symposium on Operating Systems Principles*, pages 309–324.

Wang, Y., Alvisi, L., and Dahlin, M. (2012). Gnothi: Separating data and metadata for efficient and available storage replication. In *the USENIX Annual Technical Conference*, pages 413–424.

Wu, Z., Butkiewicz, M., Perkins, D., Katz-Bassett, E., and Madhyastha, H. V. (2013). Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. In *Proceedings of the 24th Symposium on Operating Systems Principles*, pages 292–308.

ResiDI: Um Sistema de Decisão Inteligente para Infraestruturas Residenciais via Sensores e Atuadores Sem Fio

Geraldo P. R. Filho[1], **Jó Ueyama**[1], **Bruno S. Faiçal**[1]
Daniel L. Guidoni[2], **Leandro A. Villas**[3]

[1]Instituto de Ciência Matemáticas e de Computação – ICMC
Universidade de São Paulo – USP
São Carlos – SP – Brasil

[2]Universidade Federal de São João del-Rei – UFSJ
São João del-Rei – MG – Brasil

[3]Instituto de Computação – Universidade Estadual de Campinas – UNICAMP
Campinas – SP – Brasil

{geraldop, joueyama, bsfaical}@icmc.usp.br
guidoni@ufsj.edu.br, leandro@ic.unicamp.br

Abstract. *This work proposes an intelligent decision system for a residential infrastructure based on wireless sensors and actuator networks, called ResiDI. In order to achieve this, the proposal is based on the development of a neural network that aims to increase the accuracy in the decision-making, and a temporal correlation mechanism that aims to reduce the energy consumption of the network. When compared with an approach adopted in the literature, both simulated and real results show that ResiDI reduces the energy consumption of its network, while maintaining a degree of accuracy in the decision-making.*

Resumo. *Este trabalho propõe um sistema de decisão inteligente para infraestruturas residenciais usando uma rede de sensores e atuadores sem fio, nomeado como ResiDI. Para isso, a proposta baseia-se no desenvolvimento de uma rede neural que visa aumentar a precisão nas tomadas de decisões, e um mecanismo de correlação temporal que visa reduzir o consumo de energia da rede. Quando comparado com uma abordagem da literatura, os resultados simulados e reais mostram que o ResiDI reduz o consumo de energia da sua rede, enquanto mantêm uma precisão nas tomadas de decisão.*

1. Introdução

Nos últimos anos, um dos principais desafios globais está relacionado com a eficiência energética, sendo o desperdício de energia um dos fatores a ser destacado. Isto ocorre, devido a crescente demanda de energia por parte das indústrias, comércios e residências, as quais utilizam a energia de maneira não controlada. Nesse cenário, o setor residencial obteve o maior crescimento, cerca de 6,9% em relação a 2012 [EPE 2013]. Esse aumento, muitas vezes desordenado, atribuído à expansão do uso dos eletrodomésticos, ocasiona um desperdício significativo no uso da energia.

O Sistema de Automação Residencial (HAS, do inglês *Home Automation System*), integrado no sistema elétrico de potência, também conhecido como *smart home*, é um dos

caminhos para prover soluções relacionadas com a eficiência energética. O HAS permite o gerenciamento dos recursos habitacionais, tais como iluminação, aquecimento, ar condicionado e aparelhos eletrônicos, levando em consideração grandezas físicas do ambiente (temperatura, presença e umidade, por exemplo) para monitorar, bem como atuar de forma inteligente em benefício dos moradores [Baraka et al. 2013, Lee et al. 2014].

Apesar dos avanços conquistados nessa área, os HASs são fortemente dependentes da rede interna, visto que esta é a base de uma casa inteligente. Há vários trabalhos [Lee and Hong 2009, Ramlee et al. 2013] que usam infraestruturas cabeadas, juntamente com sensores analógicos, para sensorear o ambiente. No entanto, tais infraestruturas são: (i) dependentes do ambiente; (ii) custosas devido a instalação da fiação que muitas vezes são escassas; e (iii) utilizam, por exemplo, um equipamento centralizador para gerenciar as informações da rede. Além disso, a falta de uma infraestrutura padronizada para atender as necessidades do usuário, representa um dos desafios a serem considerados na implantação de uma HAS.

A Rede de Sensores Sem Fio (RSSF) [Akyildiz et al. 2002] surge como uma alternativa promissora para superar as limitações mencionadas anteriormente, permitindo flexibilidade na remoção e/ou adição de componentes, facilidade em integrá-la dentro de outros *backbones*, além de ser escalável, colaborativa e de baixo custo. No entanto, as RSSFs são formadas por dispositivos passíveis (nós sensores), limitados em recursos (memória, bateria e processamento, por exemplo), que apenas coletam e disseminam dados, sem atuarem no ambiente. Nesse sentido, a combinação de sensores e de atuadores em uma mesma rede, nomeado como Redes de Sensores e Atuadores Sem Fio (RSASF) [Akyildiz and Kasimoglu 2004], aumenta o potencial das RSSFs no HAS, uma vez que as RSASFs possuem maior poder de processamento, comunicação e energia, podendo atuar no ambiente físico mediante os dados provenientes dos sensores.

Neste contexto, as RSASFs fornecem uma infraestrutura moderna e ubíqua para uma casa inteligente. Entretanto, o uso das RSASFs para monitorar e atuar como infraestrutura de controle dentro do contexto de uma HAS provoca novos desafios [Lee and Hong 2009, Han and Lim 2010, Soares et al. 2012, Ramlee et al. 2013, Wang et al. 2013, Baraka et al. 2013, Filho et al. 2013, Filho et al. 2014, de Farias et al. 2014, Lee et al. 2014], dos quais dois se destacam, a saber: (i) o desenvolvimento de soluções computacionais inteligentes, para que o processo de tomada de decisão seja realizado de maneira distribuída dentro da própria RSASF; e (ii) o prolongamento da vida útil dos nós da RSASF, mantendo um *trade-off* entre precisão das decisões e consumo de energia dos sensores, sem que haja perdas de confiabilidade na infraestrutura distribuída.

Para superar os desafios mencionados anteriormente, este trabalho propõe o ResiDI (***Resi**dential **D**istributed **I**nfrastructure*), um sistema de decisão inteligente para uma infraestrutura residencial distribuída usando sensores e atuadores sem fio. Para isso, a proposta baseia-se no desenvolvimento de uma rede neural para aumentar a precisão nas tomadas de decisões, bem como um mecanismo de correlação temporal para reduzir o consumo de energia da rede. Desta forma, o ResiDI permite a formação de uma RSASF para sensorear e atuar no processo de tomada de decisão de forma autônoma dentro da própria rede, sendo ciente no tempo de vida útil dos nós da infraestrutura e na precisão das tomadas de decisões.

Os resultados dos experimentos, evidenciam o desempenho satisfatório do ResiDI quando comparado com o CONDE [Soares et al. 2012], superando-o em diferentes cenários para as avaliações realizadas.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve estratégia utilizada para o desenvolvimento do ResiDI. A Seção 4 apresenta os resultados obtidos da avaliação de desempenho. Finalmente, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Diversos trabalhos encontrados na literatura [Lee and Hong 2009, Han and Lim 2010, Soares et al. 2012, Baraka et al. 2013, Ramlee et al. 2013, Wang et al. 2013, Filho et al. 2013, Filho et al. 2014, de Farias et al. 2014, Lee et al. 2014] têm tratado do uso de sensores nos sistemas residências. Contudo, apesar dos avanços conquistados, ainda há vários problemas nessa área. Por exemplo, a inexistência de uma solução baseada em rede neural juntamente com correlação temporal, incorporando-as no processo de tomada de decisão dentro da rede.

Os trabalhos descritos em [Han and Lim 2010, Baraka et al. 2013, Ramlee et al. 2013] propõem sistemas específicos para apenas uma aplicação como iluminação ou ar-condicionado, por exemplo. Particulamente em [Ramlee et al. 2013], é proposto um sistema de automação residencial de baixo custo voltado para os eletrodomésticos. Para tanto, uma placa de controle principal (nó *sink*) é desenvolvida para gerenciar a comunicação entre os eletrodomésticos e os moradores. No entanto, as principais limitações desses tipos de sistemas são a sua centralização e não correlacionar os dados coletados.

Embora haja trabalhos sobre a composição de mais de uma aplicação, tais trabalhos utilizam sensores apenas como infraestrutura para a coleta de dados, como propostos por [Wang et al. 2013, Filho et al. 2014]. Por isso, essas soluções não aproveitam os benefícios que uma RSASF possui para a tomada de decisão. Em [Lee et al. 2014], é proposto um sistema sensível ao contexto para ambientes residências ubíquos. Para isso, utilizou-se dados providos dos sensores para analisar padrões e gerar regras, inferindo assim as preferências dos moradores. Tal sistema diferencia-se deste trabalho por depender de *feedbacks* dos moradores para a tomada de decisão, ferindo um dos princípios da computação ubíqua que defende as interfaces invisíveis.

Outros trabalhos relacionados que merecem destaques são propostos por [Soares et al. 2012, de Farias et al. 2014]. Convém salientar que o trabalho apresentado por [de Farias et al. 2014], é um complemento do trabalho realizando anteriormente em [Soares et al. 2012]. Em ambos os trabalhos, os autores propõem um sistema de CONtrole e DEcisão (CONDE) com o intuito de automatizar o processo de tomada de decisão em um ambiente usando uma RSASF. O CONDE é desenvolvido para coletar e manipular os dados sensoreados, processando-os dentro da própria infraestrutura. A diferença do [de Farias et al. 2014] para o [Soares et al. 2012], é a proposta de um subsistema de inferência de nível múltiplo para proporcionar um consenso no processo de apoio à tomada de decisão. Contudo, deve-se ressaltar que ambos os trabalhos possuem duas limitações, a saber: (i) regras pré-estabelecidas para a tomada de decisão; e (ii) não há uma abordagem que explore a correlação para a redução do consumo de energia da

infraestrutura. Por esses motivos, a seguir será apresentado o ResiDI.

3. Sistema de Decisão Inteligente para Infraestruturas Residenciais

Nesta seção é apresentado o ResiDI, uma proposta para a tomada de decisão em uma infraestrutura residencial distribuída usando sensores e atuadores sem fio. Para tanto, uma solução baseada em rede neural para atuar no processo de tomada de decisão, bem como um mecanismo de correlação temporal para maximizar a eficiência energética da infraestrutura foram desenvolvidas. Com isto, o principal objetivo do ResiDI é manter uma precisão na tomada de decisão, e prover uma redução no consumo de energia nos nós da rede. Para alcançar tais objetivos, considerou-se os seguintes papéis:

- *Nó Sensor* (Rótulo 1, Figura 1). Coleta dados de diferentes grandezas físicas do ambiente (por exemplo, temperatura e presença), transmitindo-os para o nó decisor (linha tracejada, Figura 1). Além disso, é executado um mecanismo de correlação temporal ponderada.
- *Nó Decisor* (Rótulo 2, Figura 1). Recebe os dados coletados dos nós sensores, processa tais dados para a tomada de decisão, e encaminha a decisão para o nó atuador (linha continua, Figura 1).
- *Nó Atuador* (Rótulo 3, Figura 1). Recebe a ação a ser realizada do nó decisor e atua na aplicação desejada. Além disso, o nó atuador pode atua em diferentes aplicações (por exemplo, iluminação, ar-condicionado e aparelhos eletrônicos), as quais estão em seu alcance.
- *Nó Sink* (Rótulo 4, Figura 1). Responsável por realizar a comunicação externa e possíveis atualizações vindas fora da rede. Nesse cenário, o nó *sink* pode permanecer ativo por mais tempo, e por essa conveniência a sua implantação pode ser próximo de um roteador ou uma tomada, tendo como fonte alternativa de energia o *Power over Ethernet* ou um carregador, respectivamente.

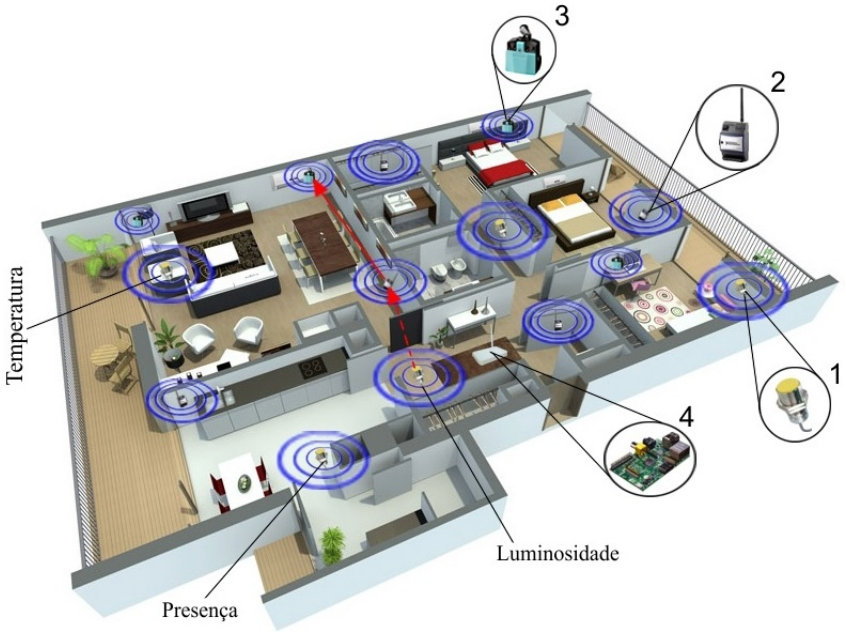


Figura 1. Exemplo de funcionamento do ResiDI.

A Figura 1 sumariza a proposta do ResiDI, o qual é baseado em sensores da plataforma MicaZ[1] que representa os nós sensores, decisores e atuadores; bem como o Raspberry Pi[2] que representa o nó *sink*. Como já é constatado que acumular mais de um papel em um nó é energeticamente ineficiente [Souza and Mateus 2006, Soares et al. 2012], as funcionalidades do ResiDI foram distribuídas em papeis. Com essa abordagem, além de deixar a infraestrutura com alta disponibilidade para os nós que a compõem, também reduz o tráfego da rede e maximiza a sua eficiência na entrega dos dados. Além disso, caso ocorra uma falha no nó atuador, o nó decisor mantém-se ativo, escolhendo outro nó atuador para finalizar a tarefa, sendo possível acionar, caso necessário, mais de um nó atuador usando apenas um nó decisor. Por outro lado, ao considerar uma infraestrutura sem divisão de papeis, com um único nó para decidir e atuar, por exemplo, caso ocorra uma falha no componente de atuação a tarefa não é finalizada com êxito. Ainda nesse cenário, para realizar uma tomada de decisão adequada, os nós necessitariam compartilhar dados entre si [Soares et al. 2012]. Com isto, além de aumentar o tráfego na rede, haveria transmissões de ações repetidas para os atuadores. Portanto, haveria a necessidade de utilizar algoritmos de sincronização, o que deixaria a proposta complexa e inviável na prática. Para o desenvolvimento do ResiDI, algumas etapas foram necessárias: (i) a infraestrutura, Subseção 3.1; e (ii) a tomada de decisão, Subseção 3.2. Tais etapas são apresentadas a seguir.

3.1. Infraestrutura do ResiDI

Esta seção apresenta como foi definida a infraestrutura do ResiDI. Para tanto, dividiu-se a solução desta etapa em três fases. Na primeira, os nós cooperam entre si para a construção da tabela de roteamento. Na segunda, os dados são sensoreados e a correlação temporal é aplicada. Finalmente, na terceira, a tomada de decisão é transmitida para o nó atuador.

3.1.1. Construção da Tabela de Roteamento

A tabela de roteamento é construída contendo as distâncias dos nós sensores vizinhos. A obtenção da distância como métrica para encaminhar as mensagens é realizada mediante a intensidade do sinal recebido (RSSI). A utilização do RSSI é preferível em RSASF por ser uma técnica simples, eficaz e por não necessitar de *hardware* extra. Ainda, a técnica do RSSI como métrica de distância vem sendo aplicada em diversos cenários, que vão desde ambientes *indoor* [Pessin et al. 2014] até *outdoor* [Lima et al. 2013].

Para facilitar o entendimento da técnica, a Figura 2 ilustra como é estimada a distância entre os nós. Para isso, quando um nó recebe uma mensagem, a força do sinal captada por ele é utilizada para estimar a distância do nó que transmitiu a mensagem. Assim, por exemplo, quanto mais próximo o nó sensor está do nó decisor, maior é a potência do sinal para o nó sensor e consequentemente menor é a sua distância. Desta forma, os nós sensores armazenam em sua tabela de roteamento as distâncias dos nós decisores para encaminhar os dados sensoreados. Já os nós decisores armazenam as distâncias dos nós atuadores. Em razão disso, os nós comunicam-se em um único salto. Deve-se ressaltar que a propagação de sinal foi baseada no modelo *Two-Ray Ground*. Tal modelo além de assumir que o sinal transmitido não é recebido por apenas um caminho, considera a possibilidade de reflexões.

[1]Micaz, http://www.memsic.com/wireless-sensor-networks/

[2]Raspberry Pi, http://www.raspberrypi.org/

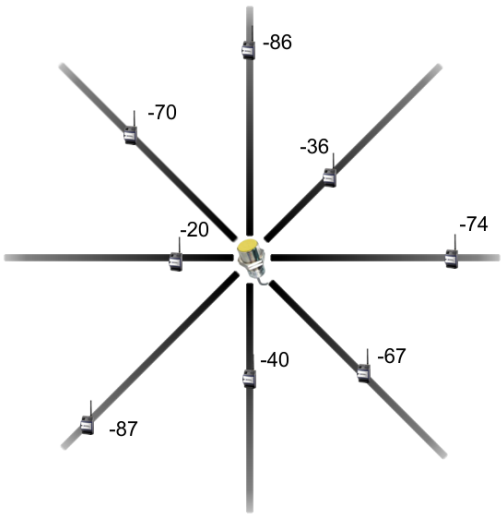


Figura 2. Potência do sinal em dBm decrescendo à medida que a intensidade do sinal se propaga na rede, tornando a distância inversamente proporcional a intensidade do sinal.

3.1.2. Comunicação do nó Sensor com o nó Decisor

Para sensorear os dados do ambiente e transmiti-los para o nó decisor com maior eficiência energética, foi necessário propor um mecanismo de correlação temporal ponderada, apresentado no Algoritmo 1. Por correlação temporal, entende-se que os dados coletados são correlacionados ao tempo, comparando o padrão de leitura atual com as leituras anteriores para transmitir, ou não, o dado coletado. Desta forma, é possível reduzir o consumo de energia da rede, eliminando transmissões desnecessárias. Para tanto, esse processo ocorre depois dos nós sensores descobrirem os nós decisores mediante o RSSI, descrito na Seção 3.1.1. Após a descoberta, cada nó n mantém um conjunto S das leituras reportadas R_{old}, $\{R_{old}, old \in S\}$, em seu *buffer* (Linha 1, Algoritmo 1). Com o intuito de oferecer prioridade aos dados de leitura mais novos, atribui-se um conjunto de pesos $P(w) = \{1, 2, 3, ..., w\}$ aos índices mais recentes de acordo com o tamanho do conjunto S (Linha 2, Algoritmo 1). Com isto, é possível definir um *threshold*: $\frac{\sum_{w=1}^{S} R_{old} * P_w}{\sum P_w}$ (Linha 5, Algoritmo 1). Assim, a cada leitura realizada R_{new}, o nó verifica se a R_{new} é maior que o *threshold* (Linha 6, Algoritmo 1). Caso a condição seja satisfeita, o valor sensoreado (R_{new}) é transmitido para o nó decisor, utilizando a tabela de roteamento (Linha 7, Algoritmo 1). Por fim, a R_{new} é adicionada no conjunto S com maior prioridade, enquanto que o valor (R_{old}) de menor prioridade é suprimindo (Linha 9, Algoritmo 1).

3.1.3. Comunicação do nó Decisor com o nó Atuador

Os nós decisores decidem e coordenam as ações na residência. Ainda, é nessa etapa que os dados coletados são associados às aplicações. Para isso, modelou-se uma rede neural (descrito com mais detalhes na Seção 3.2), a qual é capaz de dar sentido mais amplo para os dados sensoreados.

No Algoritmo 2, é descrito o funcionamento do nó decisor. Antes de inicializar tal fase, cada nó decisor descobre os nós atuadores mediante o RSSI. Ainda nessa fase, são obtidas as informações dos *Mac Address* dos atuadores, bem como suas respectivas

Algoritmo 1: Mecanismo de correlação temporal e transmissão dos dados monitorados

```
1  S_n ← GetR_old() ;                                   // Leituras reportadas no buffer
2  P_w ← SetIndex(S_n) ;                                // Atribui pesos aos dados reportados
3  R_new ← GetNew() ;                                   // Leitura atual
4  para cada n ∈ S_n faça
5      Threshold ← (∑_{w=1}^{S} R_old*P_w) / (∑ P_w) ;   // Calcula a correlação temporal ponderada
       // verifica se o R_new passa o Threshold
6      se R_new > Threshold então
7          ToSend(R_new, TabelaRoteamento); ;            // Envia dados para o nó decisor
8      fim
       // Atualiza o Threshold adicionando o R_new e suprimindo o R_old
9      S_n ← UpdateS(S_n, R_new);
10 fim para
```

aplicações que eles podem atender. A próxima fase do Algoritmo 2 está relacionada com o processo de tomada de decisão dentro do próprio nó decisor. Esse processo só é iniciado quando o nó decisor recebe um conjunto de dados dos nós sensores mediante uma correlação temporal. O nó decisor ao receber esses dados toma a decisão com base na proposta de uma rede neural já treinada, descrito na próxima seção. Desta forma, a informação da aplicação e sua respectiva decisão são obtidas (Linha 3, Algoritmo 2). Com tais informações, encaminha-se a decisão para o nó atuador. Deve-se ressaltar, portanto, que a decisão é transmitida para o nó atuador mais próximo (Linhas 4 a 6, Algoritmo 2) com o intuito de poupar energia residual do nó decisor.

Algoritmo 2: Coordena as tomadas de decisões e transmiti para o nó atuador

```
1  R_n ← ReadSet() ;                                    // Conjunto de dados dos nós sensores
2  para cada d ∈ R_n faça
3      DecisionMaking ← RedeNeural(R_n) ;                // toma a decisão
       // verifica tomada de decisão
4      se DecisionMaking.Act = True então
5          ToSend(DecisionMaking, TabelaRoteamento) ;    // Envia a decisão para o nó atuador
6      fim
7  fim para
```

3.2. Mecanismo para a Tomada de Decisão do ResiDI

O modelo proposto para a tomada de decisão tem como objetivo o aprendizado das aplicações mediante uma rede neural do tipo *Multi-Layer Perceptron*. A escolha da rede neural é justificada por alguns motivos, sendo os principais: (i) capacidade de generalização na tomada de decisão; (ii) auto-aprendizado baseado apenas nos exemplos históricos; e (iii) imunidade a ruídos para dados reais. Além disso, ao contrário de outros métodos, a rede neural possibilita uma adaptação as condições que possam ocorrer e com uma maior estabilidade na tomada de decisão.

Para facilitar o entendimento da rede neural, a Figura 3 ilustra a topologia da rede neural e como os dados de entrada (isto é, dados provenientes dos nós sensores) são utilizados para a tomada de decisão. Para tanto, a topologia da rede foi modelada com as seguintes configurações: (i) camada de entrada com sete neurônios, os quais correspondem aos dados transmitidos pelos nós sensores; (ii) camada escondida com quinze neurônios, os quais representam a capacidade de aprendizagem para a tomada de decisão;

e (iii) camada de saída com três neurônios, representando a tomada de decisão a ser enviada para o nó atuador e sua aplicação, sendo esta composta por uma combinação de neurônios, uma vez que é possível ter mais de uma decisão para diferentes aplicações.

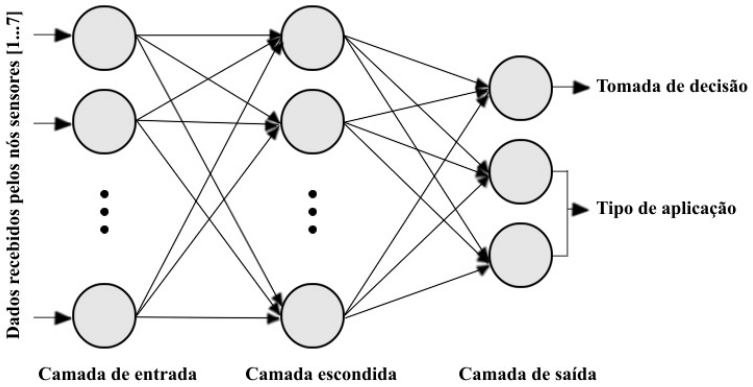


Figura 3. Topologia da rede neural modelada para a tomada de decisão.

Deve-se ressaltar, portanto, que com a rede treinada e transmitida para os nós decisores, é possível dar sentido mais amplo para a tomada de decisão por meio das grandezas físicas compartilhadas. Isto é, um único sensor de presença pode ser utilizado para o sistema de ar condicionado e iluminação. Desta forma, os dados coletados não são associados apenas a um evento. Por exemplo, quando o sensor de presença detecta ausência de moradores, o sistema de ar condicionado pode desligar ou aumentar a temperatura, enquanto, paralelamente, o sistema de iluminação pode ser desligado.

É fundamental destacar que para treinar a rede, optou-se por realizar no nó *sink*. Nesse caso, utilizou-se o Raspberry Pi por ser uma plataforma de prototipagem aberta e de baixo consumo de energia. No nó *sink*, encontra-se a base de dados utilizada para treinar a rede, descrita na Seção 3.2.1. Após o treinamento, deve-se ressaltar que o nó *sink* transmite a rede treinada para os nós decisores. Assim, não há *overhead* (isto é, processamento e/ou armazenamento em excesso) no ResiDI. Com isto, é possível re-treinar a rede, caso necessário, mediante o comportamento do(s) morador(es), sem prejuízos na eficiência energética do sistema. A seguir é apresentada a base de dados utilizada e como a rede foi treinada.

3.2.1. Base de Dados

Com o intuito de obter dados mais confiáveis para os experimentos, utilizou-se dados de um ambiente real para gerar os resultados, adquiridos em [Filho et al. 2014, Bache and Lichman 2013]. A coleta de dados foi realizada durante 3 meses em [Filho et al. 2014] e 40 dias em [Bache and Lichman 2013], ambos em um ambiente residencial. Com isso, uma base de dados foi modelada com base nos sensores de temperatura, presença, energia e um sensor lógico para captar o horário de utilização e o estado do equipamento (desligado, ligado e *standby*). As aplicações utilizadas na base de dados foram iluminação e eletrodoméstico (televisão, computador e geladeira). Deve-se ressaltar que as atividade de atuação estão relacionadas em ligar e desligar as aplicações mediante a preferência do morador.

Após a formação da base de dados, oito partições das amostras do conjunto de

dados foram feitas. Cada partição contém um conjunto de dados de 1 e 2 semanas. Em seguida, utilizou-se o algoritmo *back-propagation* para treinar a rede. Além disso, para evitar o problema de mínimo local, bem como estimar o quão correto o ResiDI será executado na prática, a técnica *k-fold cross-validation* com k=4 foi utilizada para separar o conjunto de treino e teste em cada partição de maneira disjunta. Desta forma, utilizou-se um subconjunto para teste e os outros três para treino. A seguir é apresentada uma avaliação de desempenho para validar o ResiDI.

4. Avaliação de Desempenho e Metodologia

Nesta seção, o desempenho do ResiDI é avaliado, comparando-o com o CONDE. A escolha do CONDE deve-se ao seu bom desempenho quando comparado com outra solução da literatura [Feng et al. 2008]. Além disso, é um dos trabalhos que mais se assemelha com a proposta deste artigo.

A validação do ResiDI foi dividida em dois estágios, a saber: (i) a avaliação de desempenho para determinar a sua capacidade de atuar no processo de tomada de decisão; e (ii) a avaliação de desempenho da infraestrutura da rede em termos de consumo de energia e tempo de resposta. Para o primeiro estágio, experimentos reais, utilizou-se o nó *sink* (Raspberry Pi) para treinar a topologia da rede neural que transmite a rede treinada para o nó decisor (MicaZ). Após o termino do primeiro estágio, utilizou-se o Network Simulator 3[3] para realizar a simulação da infraestrutura do ResiDI, tendo como parâmetros de simulação os nós da plataforma MicaZ. A seguir os resultados obtidos são apresentados.

4.1. Avaliação de desempenho do ResiDI na tomada de decisão

Esta subseção avalia o desempenho do ResiDI em determinar a sua capacidade na tomada de decisão. Para isso, considerou-se as seguintes medidas de desempenho: (i) sensibilidade; (ii) precisão; e (iii) especificidade. Tais medidas são obtidas mediante uma matriz de confusão, como ilustrada na Figura 4.

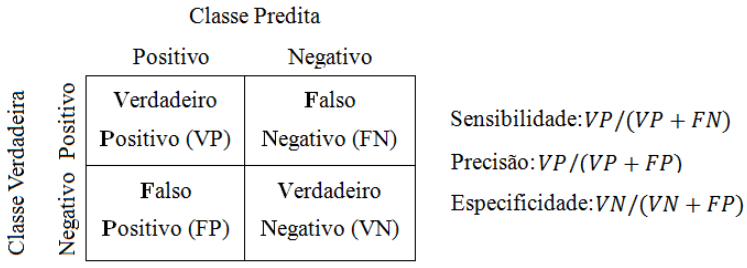


Figura 4. Medidas de desempenho calculadas a partir da matriz de confusão.

Os conjuntos de parâmetros estabelecidos para realizar a avaliação de desempenho são sumarizados na Tabela 1. O sistema é avaliado com o uso dos métodos (ResiDI e CONDE), e com a mudança do *dataset* (1 semana e 2 semanas). Cada conjunto de parâmetros foi executado 12 vezes. Com o intuito de verificar a adequação a normalidade dos resultados apresentados na Figura 5, utilizou-se o *Shapiro-Wilk normality test*, o qual obteve os p-valores apresentados na Tabela 2. Com isso, notou-se que com exceção

[3]Network Simulator 3 - NS3, http://www.nsnam.org/

de um p-valor (E3 precisão, Tabela 2), os demais conjuntos tem a hipótese de adequação à normalidade aceita considerando 95% de confiança. Por isso, utilizou-se o *t-student* como método paramétrico de comparação entre os conjuntos. Os resultados são apresentados na Figura 5, com 95% de confiança.

Tabela 1. Conjunto de parâmetros escolhidos para serem avaliados.

Experimentos	Métodos	*Dataset*
E1	ResiDI	1 semana
E2	ResiDI	2 semanas
E3	CONDE	1 semana
E4	CONDE	2 semanas

Tabela 2. P-valores resultades do *Shapiro-Wilk normality test*.

Experimentos	Shapiro-Willk normality test		
	Sensibilidade	Precisão	Especificidade
E1	0,472	0,8927	0,5031
E2	0,4322	0,2301	0,7154
E3	0,3433	**0,01977**	0,1622
E4	0,5224	0,754	05234

No gráfico da Figura 5, é apresentado a porcentagem dos resultados obtidos em função das medidas de desempenho para o *dataset* de 1 e 2 semanas. O resultado destaca que houve estatisticamente diferença significativa entre o ResiDI e o CONDE, independente do período monitorado (experimentos {(E1, E3), (E2, E4)} da Figura 5). Através desses resultados, notou-se que o ResiDI possui um desempenho superior em relação ao CONDE quando considerado o *dataset* de 1 semana. Além disso, tal aumento é mais acentuado quando considerado o *dataset* de 2 semanas. Esta situação é derivada do treinamento da rede, a qual possui mais exemplos para a sua auto-aprendizagem. Isto é, o ResiDI aprende a tomar decisões pela sua própria experiência, a qual pode ser obtida mediante os históricos comportamentais dos moradores (revisite a Seção 3.2).

Outro aspecto importante, notado no resultado da Figura 5, está relacionado com a estabilidade do ResiDI quando comparado com o CONDE. Isto é identificado pela amplitude interquartil, a qual indica o grau de dispersão dos dados gerados. Isso faz sentido, posto que o ResiDI generaliza as suas decisões com base em exemplos históricos, separando as tomadas de decisões em hiperplanos. Portanto, é possível tomar decisões corretamente para instâncias nunca vistas na base de dados. Já o CONDE realiza a sua tomada de decisão mediante um espaço fixo para cada aplicação. Em virtude disso, há um alto grau de dispersão nos resultados obtidos.

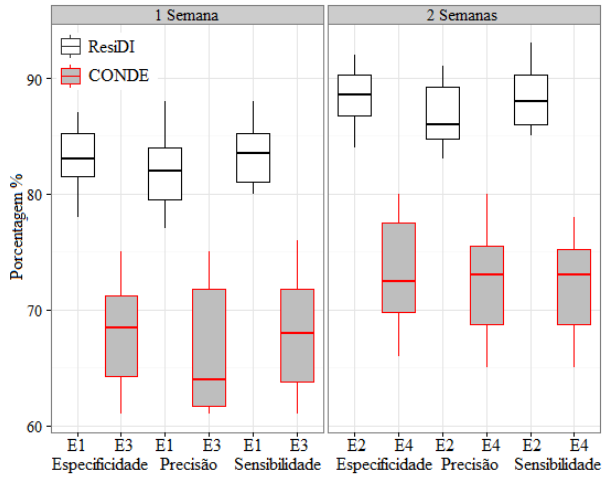


Figura 5. Análise de desempenho do ResiDI de acordo com a Tabela 1.

4.2. Avaliação de desempenho da infraestrutura do ResiDI

Esta subseção avalia o desempenho da infraestrutura do ResiDI em um ambiente simulado. Para obter um cenário mais próximo de um ambiente real, utilizou-se como entrada para a simulação os conjuntos de dados reais de 1 e 2 semanas descritos na Seção 3.2.1. Em relação a topologia da rede, adotou-se uma topologia plana para depositar os nós. Com o intuito de evitar buracos na topologia, a densidade dos nós decisores e atuadores são de 0.007 nós/m^2. Já a densidade dos nós sensores é de 0.015 nós/m^2, uma vez que é realístico admitir que há mais nós sensoreando um ambiente. O raio de comunicação dos nós é de 15 m, e o seu consumo de energia para transmitir e receber um pacote são de 0,2088 μJ/bit e 0,2256 μJ/bit, respectivamente, de acordo com a plataforma MicaZ. Além disso, adicionou-se 5% de imprecisão no RSSI. O tempo de simulação foi de uma hora, sendo o primeiro minuto para o *startup* da rede.

Os conjuntos de parâmetros estabelecidos para realizar a avaliação de desempenho são apresentados na Tabela 3. Os experimentos foram avaliados com a alteração da área monitorada (100 m^2, 225 m^2, 400 m^2 e 625 m^2), e com a troca dos métodos (ResiDI e CONDE). Como o objetivo da avaliação é determinar a capacidade do ResiDI em obter soluções energeticamente eficiente, utilizou-se como variável de resposta as métricas: (i) atraso na tomada de decisão, o qual representa o tempo gasto desde o sensoreamento até a atuação para que uma decisão seja executada; e (ii) o consumo de energia dos nós, o qual indica a quantidade de energia consumida pela infraestrutura no processo de tomada de decisão. As configurações de cada conjunto de parâmetro foram executadas 33 vezes, com uma confiança de 95% de acordo com a distribuição *t-student*. A Tabela 4 apresenta os resultados da comparação entre os conjuntos de parâmetros $\{$(E1,E5), (E2,E6), (E3,E7), (E4,E8)$\}$. Como os resultados dos p-valores foram inferiores a 0,05, todos os resultados das comparações apresentados na Figura 6 são estatisticamente diferentes.

Tabela 3. Conjunto de parâmetros escolhidos para serem avaliados.

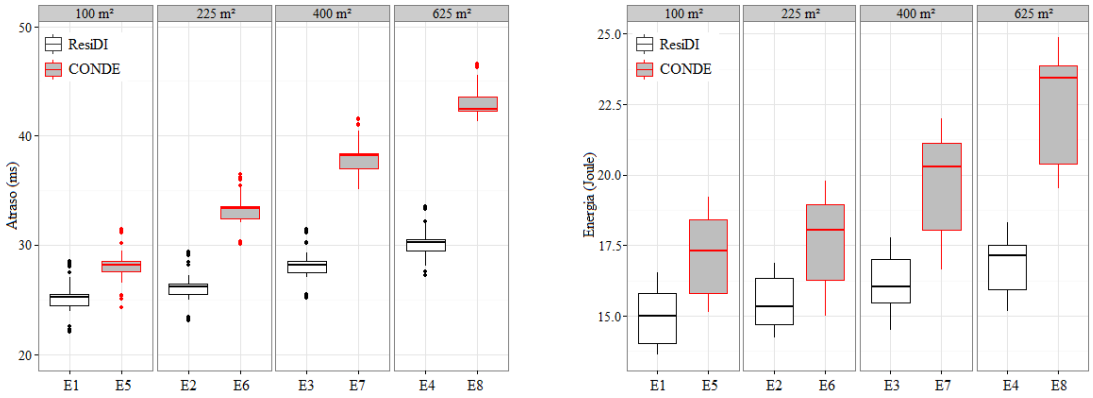
Experimento	Método	Área monitorada
E1	ResiDI	100 m^2
E2	ResiDI	225 m^2
E3	ResiDI	400 m^2
E4	ResiDI	625 m^2
E5	CONDE	100 m^2
E6	CONDE	225 m^2
E7	CONDE	400 m^2
E8	CONDE	625 m^2

Tabela 4. P-valores resultantes do *t-student*.

Comparação entre os conjuntos de parâmetros	t-student	
	Atraso	Energia
E1 E5	$\approx 0,000$	$\approx 0,000$
E2 E6	$\approx 0,000$	$\approx 0,000$
E3 E7	$\approx 0,000$	$\approx 0,000$
E4 E8	$\approx 0,000$	$\approx 0,000$

No gráfico da Figura 6a, são apresentados os resultados referentes ao atraso no processo de tomada de decisão em função da área, ilustrando uma análise comparativa entre o ResiDI e o CONDE. Nota-se, estatisticamente, que o ResiDI realiza o processo de tomada de decisão em um tempo menor quando comparado com o CONDE, em todos os experimentos. Nos experimentos E1 e E5 da Figura 6a, o processamento do ResiDI foi no mínimo 11,8% mais rápido que CONDE. Isso ocorre em virtude da formulação do hiperplano separador gerado pela rede neural, a qual não necessita consultar uma base de dados para chegar a um consenso no processo de tomada de decisão, ao contrário do CONDE. Desta forma, é possível obter respostas mais rápidas, o que influencia diretamente na redução do consumo de energia da rede. Além disso, quando há uma ampliação na área de interesse, observou-se que houve diferenças significativas de 35,04% e 44,35%

para os experimentos (E3, E7) e (E4, E8) entre o ResiDI e CONDE. Em outras palavras, no ResiDI, houve uma redução mais acentuada no tempo de entrega das mensagens para a tomada de decisão. Portanto, em relação a métrica atraso, o ResiDI possui um desempenho superior em relação ao CONDE, apresentando melhor robustez que a solução da literatura.



(a) Atraso

(b) Consumo de energia

Figura 6. Análise de desempenho das métricas atraso e consumo de energia de acordo com a Tabela 3.

Com base no conjunto de parâmetros propostos, o gráfico da Figura 6b apresenta os resultados relativos à métrica consumo de energia para cenários com diferentes áreas monitoradas. Para o cenário com uma área de 100 m^2 e 225 m^2, observa-se que há uma redução de aproximadamente 14,11% e 13,79% do consumo de energia quando as soluções propostas pelo ResiDI é utilizada e, consequentemente, um aumento no tempo de vida útil da infraestrutura. Isso ocorre devido aos dados transmitidos serem correlacionados ao tempo. Em outras palavras, somente quando há uma diferença da leitura atual com as leituras reportadas, o dado sensoreado é encaminhado para o nó desejado (revisite a Subseção 3.1.2). Além disso, deve-se salientar, portanto, que o aumento na área monitorada não afeta o desempenho do ResiDI, uma vez que o consumo de energia foi reduzido de forma significativa para 22,24% e 33,31% considerando as áreas de 400 m^2 e 625 m^2, respectivamente (experimentos {(E3, E7), (E4, E8)} da Figura 6b). Esta diferença significativa em relação ao consumo de energia faz sentido, posto que o trabalho da literatura penaliza o nó decisor por consumir mais energia, o qual é considerado um ponto de falha do sistema.

5. Conclusões e Trabalhos Futuros

Com o objetivo de atender os problemas encontrados na literatura, este artigo propôs o ResiDI, um sistema de decisão inteligente para uma infraestrutura residencial distribuída usando sensores e atuadores sem fio. O ResiDi além de manter uma precisão no processo de tomada de decisão, tem como objetivo minimizar o consumo de energia da infraestrutura. Com o intuito de comprovar a viabilidade do ResiDI, os resultados obtidos das avaliações de desempenho permitiu validar a sua eficiência. Nesse sentido, quando comparado com uma abordagem da literatura, os resultados apresentados são promissores,

sendo os principais: (i) aumento da precisão na tomada de decisão; (ii) redução no tempo de atraso para que uma decisão seja executada; e (iii) maximização do tempo de vida útil da rede. Como trabalhos futuros, planeja-se desenvolver soluções de agregação de dados para sistemas residências, investigando o seu impacto em termos de eficiência energética. Além disso, pretende-se investigar modelos de privacidade, uma vez que o sistema de automação residencial podem revelar informações detalhadas do dia-a-dia do morador.

6. Agradecimentos

Os autores gostariam de agradecer o apoio financeiro concedido da FAPESP (processo 2014/06330-5) para o bolsista de doutorado MSc. Geraldo P. R. Filho. Ainda, os autores também agradecem ao CNPq (processos 473493/2013-6 e 8486332/2013-6), CAPES (processo DS-6706658/D) e FAPESP (processos 2012/22550-0 e 2014/19076-0) por financiarem partes dos seus projetos de pesquisas. Por fim, Jó Ueyama agradece o *Office of Naval Research Global* pelo suporte financeiro em sua pesquisa.

Referências

Akyildiz, I. F. and Kasimoglu, I. H. (2004). Wireless sensor and actor networks: research challenges. *Ad Hoc Networks*, 2(4):351 – 367.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). A survey on sensor networks. *Communications magazine*, 40(8):102–114.

Bache, K. and Lichman, M. (2013). UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science [http://archive.ics.uci.edu/ml/machine-learning-databases/00235/] visited on March 22, 2015.

Baraka, K., Ghobril, M., Malek, S., Kanj, R., and Kayssi, A. (2013). Low cost arduino/android-based energy-efficient home automation system with smart task scheduling. In *Comput. Intel., Communicat. Syst. and Networks*, pages 296–301. IEEE.

de Farias, C., Soares, H., Pirmez, L., Delicato, F., Santos, I., Carmo, L. F., de Souza, J., Zomaya, A., and Dohler, M. (2014). A control and decision system for smart buildings using wireless sensor and actuator networks. *Transactions on Emerging Telecommunications Technologies*, 25(1):120 – 135.

EPE (2013). Retrieved from: http://www.epe.gov.br/ResenhaMensal/20131030_1.pdf. visited on January 6, 2014.

Feng, M.-W., Wen, S.-L., Tsai, K.-C., Liu, Y.-C., and Lai, H.-R. (2008). Wireless sensor network and sensor fusion technology for ubiquitous smart living space applications (invited paper). In *Universal Communication, 2008. ISUC '08. Second International Symposium on*, pages 295–302. IEEE.

Filho, G. P. R., Ueyama, J., Villas, L. A., Pinto, A. R., Goncalves, V. P., Pessin, G., Pazzi, R. W., and Braun, T. (2014). Nodepm: A remote monitoring alert system for energy consumption using probabilistic techniques. *Sensors*, 14(1):848–867.

Filho, G. P. R., Ueyama, J., Villas, L. A., Pinto, A. R., Goncalves, V. P., and Seraphini, S. (2013). An intelligent approach for improving energy efficiently in smart grids. In *International Conference on Wireless Networks (ICWN)*, volume 12, pages 227 – 233.

Han, D.-M. and Lim, J.-H. (2010). Smart home energy management system using ieee 802.15.4 and zigbee. *Consumer Electronics, IEEE Transactions on*, 56(3):1403–1410.

Lee, J. H., Lee, H., Kim, M. J., Wang, X., and Love, P. E. (2014). Context-aware inference in ubiquitous residential environments. *Computers in Industry*, 65(1):148 – 157.

Lee, W. S. and Hong, S. H. (2009). Implementation of a knx-zigbee gateway for home automation. In *Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on*, pages 545–549. IEEE.

Lima, M. M., de Oliveira, H. A. B. F., Nakamura, E. F., and Loureiro, A. A. F. (2013). Roteamento e agregacao de dados baseado no rssi em redes de sensores sem fio. In *XXXI Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos (SBRC)*, pages 555 – 568.

Pessin, G., Osorio, F. S., Ueyama, J., Wolf, D. F., Moioli, R. C., and Vargas, P. A. (2014). Self-localisation in indoor environments combining learning and evolution with wireless networks. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 661 – 666. ACM.

Ramlee, R., Othman, M., Leong, M., Ismail, M., and Ranjit, S. (2013). Smart home system using android application. In *Information and Communication Technology (ICoICT), 2013 International Conference of*, pages 277 – 280, Bandung. IEEE.

Soares, H., Pirmez, L., Delicato, F., and de Farias, C. (2012). Conde: Um sistema de controle e decisao para edificios inteligentes usando redes de sensores e atuadores sem fio. In *XXX Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos (SBRC)*, pages 117 – 130.

Souza, F. S. H. and Mateus, G. R. (2006). Um modelo de otimizacao para o problema de atribuicao de papeis em redes de sensores sem fio. In *XXXVIII Simposio Brasileiro de Pesquisa Operacional*, pages 1725 – 1736.

Wang, M., Zhang, G., Zhang, C., Zhang, J., and Li, C. (2013). An iot-based appliance control system for smart homes. In *Intelligent Control and Information Processing (ICICIP), 2013 Fourth International Conference on*, pages 744 – 747. IEEE.

Anatomia do Ecossistema de Pontos de Troca de Tráfego Públicos na Internet do Brasil

Samuel Henrique Bucke Brito[1], **Mateus Augusto Silva Santos**[1], **Ramon dos Reis Fontes**[1], **Danny Alex Lachos Perez**[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

{shbbrito,msantos,ramonrf,dlachosp,chesteve}@dca.fee.unicamp.br

Abstract. *For a long time many efforts were carried to increase the understanding of the complex Internet ecosystem. Internet eXchange Points have been targets since they represent a microcosm of Internet diversity. The contribution of this paper is the first empirical analysis of the IXP ecosystem in Brazil, compiling information that comprehend the characterization of members participating of these environments, including the build of AS-level connectivity graphs. By analysing the topologies we found the average adjacency of IXP's members (vertices degree), the isolated and accumulated depth of paths in advertised routes (AS-PATH), the density of peers to highlight the peering potential within IXPs, the accounting of traffic engineering policies related to AS-Prepend and k-clique communities. Another contribution is that our dataset with more than 2.5 GB is shared with the academic community.*

Resumo. *Há vários anos esforços foram conduzidos na busca por uma melhor compreensão do complexo ecossistema da Internet. Os Pontos de Troca de Tráfego (PTT) têm sido alvos constantes de pesquisas dessa natureza porque representam um microcosmo da diversidade da Internet. Este trabalho traz como contribuição a primeira análise do ecossistema de PTTs em operação no Brasil, compilando informações que compreendem desde a caracterização dos tipos de membros desses ambientes até a construção dos respectivos grafos de conectividade (em nível de Sistema Autônomo). A partir das topologias foi possível identificar a quantidade média de adjacências dos membros dos PTTs (grau dos vértices), a profundidade isolada e acumulada dos caminhos que compõem as rotas anunciadas (AS-PATH), a densidade de pares que determina o potencial de peering nos PTTs, a ocorrência de políticas de engenharia de tráfego do tipo AS-Prepend e comunidades k-clique. Outra contribuição é que o dataset com mais de 2,5 GB de informações levantadas para realização desta análise é compartilhado com a comunidade acadêmica.*

1. Introdução

Uma forma de promover o desenvolvimento da Internet em termos de conectividade e desempenho é através da implantação dos chamados Pontos de Troca de Tráfego (PTT),

do inglês *Intenet eXchange Point* (IXP). Por possuírem dezenas ou centenas de Sistemas Autônomos (SA) conectados, atualmente os PTTs têm um papel crítico no ecossistema da Internet. O OpenIX[1] e o Euro-IX[2] são exemplos de iniciativas dos Estados Unidos e da Europa, respectivamente, que promovem o desenvolvimento de PTTs. Outro exemplo de destaque é o AMS-IX[3], localizado na Holanda, que possui mais de 600 membros e atualmente é o maior PTT do mundo. No cenário nacional o projeto PTTMetro[4] (ou PTT.br) contempla todos os PTTs públicos em operação, sendo o PTT de São Paulo (PTT-SP) o maior do Brasil (e da América Latina) com registros médios de troca de tráfego da ordem de 300 Gbps e picos de 500 Gbps. Atualmente o PTT.br está entre os dez que mais trocam tráfego no mundo, sendo o quinto maior em número de participantes.

Muitas empresas justificam seus interesses em *peering* nos PTTs pelos benefícios de desempenho e segurança [Norton 2014]. Em termos de desempenho, sabe-se que o tráfego de vídeo já representa 50% de todo o tráfego da Internet e continua crescendo, o que faz o *peering* em PTTs ideal para melhorar a distribuição desse conteúdo nas proximidades dos clientes finais, já que melhora a latência ao minimizar o diâmetro entre os SAs. Durante períodos de congestionamento no tráfego de trânsito decorrente de ataques de negação de serviço, o tráfego direto através de *peering* em PTTs é separado e não sofre das vulnerabilidades do tráfego de trânsito, o que implica em mais segurança.

Exceto para aqueles que participam do ecossistema dos PTTs, é pouco disseminado o conceito de *peering* remoto, prática que está se tornando cada vez mais comum porque permite que grandes empresas e provedores de conteúdo possam ser membros dos PTTs regionais sem os altos custos dos roteadores, do espaço físico dos equipamentos (*colocation space*) e de recursos humanos. Esse crescimento implica em mais membros nos PTTs que, por sua vez, implica em mais troca de tráfego e mais rotas no ecossistema, uma dinâmica que vem aumentando o valor estratégico dos PTTs.

Nos últimos anos foram realizados vários esforços na busca por uma melhor compreensão do complexo ecossistema da Internet [Haddadi and Bonaventure 2013], principalmente através de análises baseadas em dados públicos como: (*i*) informações obtidas em entrevistas ou bancos de dados, (*ii*) ferramentas do plano de dados (ex.: *traceroute*) e (*iii*) tabelas BGP do plano de controle disponíveis em servidores de rotas [Giotsas et al. 2014, Lodhi et al. 2014, Schlinker et al. 2014, Giotsas et al. 2013, Luckie et al. 2013, Ager et al. 2012, Labovitz et al. 2010]. Os PTTs têm sido alvos de pesquisas dessa natureza porque representam um microcosmo da diversidade da Internet [Chatzis et al. 2013], tendo membros que contemplam as grandes operadoras de trânsito, provedores de acesso, provedores de conteúdo, instituições públicas e empresas privadas.

No entanto esses trabalhos focaram em um único PTT de grande porte ou em poucos. Esse artigo é inovador por dois motivos principais: (*i*) traz um estudo do ecossistema de PTTs focado exclusivamente na realidade do Brasil e (*ii*) trata-se da primeira análise envolvendo todos os PTTs que existem em operação em um país de tamanho continental. Esse estudo focado no Brasil é conveniente de ser feito porque no país existe uma política pública nacional de gestão integrada dos PTTs, sob responsabilidade do Núcleo de Informação e Coordenação do Ponto BR (NIC.br), a autoridade nacional vinculada à autoridade regional da Internet na América Latina e Caribe (LACNIC).

[1] http://www.open-ix.org [2] https://www.euro-ix.net [3] https://ams-ix.net [4] http://ptt.br

Esse trabalho traz como principal contribuição a primeira análise do ecossistema de PTTs em operação no Brasil, compilando informações que compreendem desde a caracterização dos tipos de membros desses ambientes até a construção dos respectivos grafos de conectividade (em nível de SA) dos PTTs públicos do Brasil. A partir das topologias foi possível identificar a quantidade média de adjacências dos membros dos PTTs (grau dos vértices), a profundidade isolada e acumulada dos caminhos que compõem as rotas anunciadas (AS-PATH), a densidade de pares que determina o potencial de *peering* nos PTTs, a ocorrência de políticas de engenharia de tráfego (AS-Prepend) e a identificação de comunidades do tipo k-clique.

Outra contribuição importante é que o *dataset* com mais de 2,5 GB de informações levantadas para realização dessa análise é compartilhado com a comunidade acadêmica interessada em estudos dessa natureza e pode ser acessado através da página do nosso grupo de pesquisa[5]. Nesse amplo *dataset* a comunidade acadêmica pode encontrar a relação dos 26 PTTs do projeto PTTMetro e caracterização dos participantes, além de planilhas, grafos, gráficos e scripts elaborados no processo de análise dos dados.

2. Background

Desde 1987 até 1994 a operação do backbone da rede NSFNET era responsabilidade da *National Science Foundation* (NSF) que, a partir de 1992, deu início a um plano para transferir a operação do núcleo da Internet para o setor privado. Foi no contexto desse novo ecossistema comercial que surgiram três elementos importantes: (*i*) *Network Service Providers* (NSP), responsáveis pela operação do backbone; (*ii*) *Network Access Points* (NAP), para transportar tráfego entre os NSPs a partir de locais espalhados nos EUA; e (*iii*) *Routing Arbiter* (RA), para coletar e propagar informações de roteamento nos NAPs (similar aos modernos *Route Servers*). A concepção dos PTTs nasceu ainda nessa época, uma vez que os NAPs foram criados para ser um ponto físico de conexão de vários NSPs. Ao longo dos anos os NAPs enfraqueceram porque eram mantidos por grandes operadoras com interesses próprios, o que motivou a desconexão das demais operadoras.

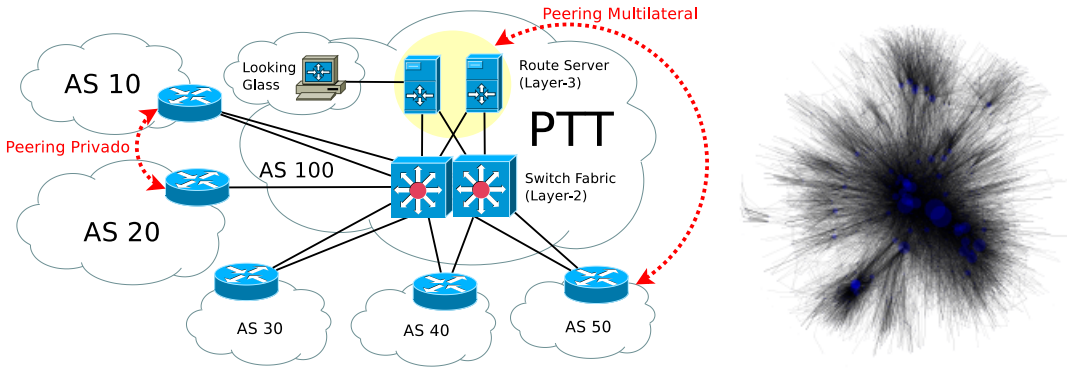
As grandes operadoras somente tinham interesse de se conectar diretamente (*peering*) com outras de mesmo nível, demonstrando pouco (ou nenhum) interesse no *peering* aberto com outras operadoras menores. Em 1999 a necessidade por circuitos ponto-a-ponto entre as grandes operadoras escalava linearmente e custava caro, além do fato de que algumas vezes as concessionárias telefônicas levavam mais de um ano para entregar um circuito [Norton 2014]. Diante dessa situação as grandes operadoras perceberam que o estabelecimento de *peering* privado através de um PTT neutro era mais rápido e econômico, tornando esse modelo dominante na Internet atual.

Por definição o PTT é uma infraestrutura compartilhada que é instalada em uma região para receber a conexão de SAs - através de *peering* - com o objetivo principal de otimizar o desempenho da Internet ao manter a troca de tráfego mais localizada entre diferentes redes pertencentes a uma mesma região, diminuindo, assim, o número de saltos entres SAs próximos uns dos outros. Seu núcleo é bastante complexo em termos de quantidade de conexões porque hospeda centenas de membros, por isso requer equipamentos de alto desempenho capazes de processar altas taxas de pacotes por segundo (pps). Apesar dessa complexidade, sua arquitetura é simples de entender, uma vez que o

[5] https://github.com/intrig-unicamp/ixp-ptt-br

objetivo do PTT se resume em prover um ponto centralizado de conexão através de uma *switching-fabric* baseada em tecnologia Ethernet (camada 2).

Uma vez fisicamente conectados no PTT, seus membros podem acordar em fazer o *peering* multilateral (aberto) com todos os demais membros ou o *peering* bilateral (privado) de natureza seletiva ou restritiva, sendo que as configurações para anúncio e alcançabilidade de prefixos IP são realizadas por meio do protocolo *Border Gateway Protocol* (BGP) [RFC4271], via sessão TCP na porta 179. Para minimizar a complexidade de configuração individual do *peering* entre todos os membros através de uma topologia *full-mesh*, são instalados elementos centrais denominados servidores de rotas (RS) na infraestrutura do PTT, de maneira que um SA é capaz de trocar rotas com os demais membros através do estabelecimento de uma única sessão BGP com os RS (*peering* multilateral), conforme pode ser observado na figura 1(a).



(a) (b)

Figura 1. (a) Arquitetura do PTT (b) Exemplo de Grafo do PTT-VIX (Vitória/ES)

O modelo de negócios do PTT pode ser de natureza privada - comum nos Estados Unidos - ou aberta, como acontece na Europa e no Brasil. Na maioria dos casos, inclusive no Brasil, o objetivo principal do PTT é prover um ambiente aberto e indiscriminatório de natureza pública para estimular a colaboração e melhorar a troca de tráfego, alavancando a qualidade da Internet na sua região de operação.

2.1. Trabalhos Relacionados

Uma análise temporal interessante é apresentada em [Ager et al. 2012], baseada em nove meses de registros coletados de um dos maiores PTTs do mundo, sendo que os autores classificaram a diversidade de membros conectados. A principal descoberta desse estudo foi que a quantidade de *peerings* de um único PTT em 2012 excedeu a quantidade total de *peering* entre SAs de toda a Internet no ano de 2010.

Um estudo focado no *Slovak Internet eXchange* (SIX) é encontrado em [Restrepo and Stanojevic 2012], sob o argumento de que existem muitos esforços e medições com interesse em compreender a Internet como um todo, mas que pouco é conhecido sobre a Internet no contexto local de uma determinada região. Os autores fizeram a caracterização da evolução do ecossistema de provedores conectados no PTT da Eslováquia e do perfil de tráfego distribuído entre seus membros.

Em [Giotsas et al. 2013] é descrito um método para descoberta de *peerings* ocultos em PTTs através da mineração do atributo *community* do BGP, problema que também

é abordado em [Chen and et al 2009]. O método é particularmente interessante porque um AS pode restringir rotas anunciadas, fazendo com que a relação de AS-PATH (caminho BGP) dos *route servers* de um PTT sejam incompletas. Através da técnica os autores inferiram 206.000 *links* ponto-a-ponto a partir de 13 PTTs da Europa, número que representa quatro vezes mais *links* do que pode ser diretamente observado nas tabelas BGP.

[Giotsas et al. 2014] argumentam que tradicionalmente as relações entre SAs são classificadas em: (*i*) trânsito (*provider-to-customer)*, (*ii*) *peering* (*peer-to-peer*) e (*iii*) *sibling* (domínios "irmãos"), no entanto existem configurações avançadas que pressupõem relações híbridas ou complexas. Para compreender essas relações complexas foi expandido o algoritmo de inferência do projeto CAIDA[6] para que sua lógica pudesse analisar tabelas BGP, saídas do aplicativo *traceroute* e dados de geolocalização. Através desse novo algoritmo foi observado que 4,5% de um universo de 90.272 relações do tipo trânsito (*provide-to-customer*) eram, na realidade, relações híbridas ou complexas.

Em [Lodhi et al. 2014] foi realizado um trabalho recente de mineração de dados no *PeeringDB*[7], uma das poucas fontes públicas carregadas com dados do ecossistema de *peering* e que consiste em uma ferramenta *online* em que as redes participantes da Internet contribuem inserindo informações sobre suas políticas, volumes de tráfego e presença geográfica. Nesse estudo os autores encontraram correlações consistentes entre os dados do *PeeringDB* com medidas realizadas a partir de prefixos BGP anunciados na Internet.

Em [Gregori et al. 2011] é aplicado um estudo em grafos de SAs conectados em PTTs em que são definidas como comunidades principais aquelas que possuem maiores comunidades, sendo que os autores concluem que tais comunidades são tipicamente grandes em tamanho e possuem baixa densidade de conexão, tendendo a se conectar em nós externos à comunidade. Um estudo similar utilizando comunidades *k-dense* é encontrado em [Orsini et al. 2013], cuja conclusão mostra que comunidades com o valor máximo de k tendem a ser compostas por provedores *tier-2* ou provedores de conteúdo.

3. Metodologia

3.1. Fontes de Informação

Duas das fontes iniciais de informações advêm dos registros do projeto PTTMetro do NIC.br e da ferramenta *online* PeeringDB. Durante o processo de análise dos dados obtidos descobrimos que o PeeringDB contém informações desatualizadas acerca dos PTTs brasileiros e seus membros, o que ficou evidente ao confrontarmos seus dados com aqueles oficiais fornecidos pelo PTT.br.

A fonte mais relevante foi a compilação de um amplo *dataset* com mais de 2,5 GB de dados a partir de um trabalho exaustivo de acesso *telnet* a todos os *Looking Glasses* (LG)[8] dos PTTs brasileiros para coletar os seguintes dados: (*i*) a tabela de rotas BGP do plano de controle, (*ii*) relação de AS-PATH do BGP e (*iii*) códigos *communities*. Em 12 dos 26 PTTs **não** foi possível extrair a tabela de rotas BGP, fato confirmado pelo NIC.br, pois o acesso público à tabela BGP completa foi desativado por problemas de escalabilidade, já que os LGs não suportam a carga de consultas. Para contornar esse problema, solicitamos esses dados diretamente ao NIC.br e fomos atendidos.

[6] www.caida.org [7] http://www.peeringdb.org

A tabela 1 traz a relação dos PTTs no Brasil, destacando suas regiões de operação, a média de troca de tráfego em Gbps, a quantidade de membros participantes e os endereços para acesso público aos seus LGs, além de observações que destacam alguns problemas que enfrentamos na coleta dos dados.

Tabela 1. PTTs Públicos em Operação no Brasil (PTTMetro)

-	Cidade	Estado	Código	*Looking Glass*	Gbps	Membros	Obs
01	Americana	SP	**AME**	lg.ame.ptt.br	0,30	9	
02	Belém	PA	**BEL**	lg.bel.ptt.br	1,66	13	
03	Belo Horizonte	MG	**MG**	lg.mg.ptt.br	3,02	34	
04	Brasília	DF	**DF**	lg.df.ptt.br	2,98	24	
05	Campina Grande	PB	**CPV**	lg.cpv.ptt.br	0,20	11	
06	Campinas	SP	**CAS**	lg.cas.ptt.br	2,58	33	(*)
07	Cuiabá	MT	**CGB**	lg.cgb.ptt.br	0,00	7	(*)
08	Caxias do Sul	RS	**CXJ**	lg.cxj.ptt.br	0,07	5	(*)
09	Curitiba	PR	**PR**	lg.pr.ptt.br	14,59	58	(1)
10	Florianópolis	SC	**SC**	lg.sc.ptt.br	1,05	32	(*)
11	Fortaleza	CE	**CE**	lg.ce.ptt.br	0,87	26	
12	Goiania	GO	**GYN**	lg.gyn.ptt.br	0,38	21	
13	Lajeado	RS	**LAJ**	lg.laj.ptt.br	0,00	7	(*)
14	Londrina	PR	**LDA**	lg.lda.ptt.br	1,45	31	
15	Manaus	AM	**MAO**	lg.mao.ptt.br	0,00	7	(*)
16	Maringá	PR	**MGF**	lg.mgf.ptt.br	0,06	14	(*)
17	Natal	RN	**NAT**	lg.nat.ptt.br	0,74	12	(*)
18	Porto Alegre	RS	**RS**	lg.rs.ptt.br	6,30	100	
19	Recife	PE	**PE**	lg.pe.ptt.br	0,03	13	
20	Rio de Janeiro	RJ	**RJ**	lg.rj.ptt.br	19,83	52	(*)
21	Salvador	BA	**BA**	lg.ba.ptt.br	1,78	41	(*)
22	São Carlos	SP	**SCA**	lg.sca.ptt.br	0,00	3	(*)
23	São José dos Campos	SP	**SJC**	lg.sjc.ptt.br	0,47	11	
24	São José do Rio Preto	SP	**SJP**	lg.sjp.ptt.br	0,03	8	(*)
25	São Paulo	SP	**SP**	lg.sp.ptt.br	334,27	550	(1)
26	Vitória	ES	**VIX**	lg.vix.ptt.br	0,22	20	

(1) Há filtros no LG que estão limitando a quantidade de rotas na tabela BGP.
(*) PTT com acesso indisponível ao LG, cujos dados foram fornecidos pelo NIC.br.

3.2. Construção dos Grafos

Através das ferramentas NetworkX[9] e Neo4j[10] (banco de dados orientado a grafos), os grafos de conectividade dos PTTs foram reproduzidos tendo como entradas as matrizes de adjacência geradas a partir dos arquivos extraídos dos PTTs, e.g., figura 1(b). Os grafos devem ser interpretados tendo como referência de vértices os SAs dos caminhos BGP (AS-PATH), cujas arestas representam sessões BGP inter-domínios.

Para reprodução dos grafos dos PTTs personalizamos vários algoritmos para a ferramenta NetworkX (baseados na linguagem Python) e escrevemos um algoritmo em Java para fazer a inserção dos AS-PATH BGP no Neo4j. Em relação ao processamento dos grafos no NetworkX e no Neo4j, cabe destacar que em ambas as ferramentas o processamento dos dados consumiu em média 2 horas para cada PTT, sendo que foram utilizados

[8] *Looking Glass* (LG) é uma máquina pública que espelha todas as rotas existentes na tabela BGP dos membros dos PTTs. São comumente acessados via *telnet* pelos membros do PTT para fins de verificação de erros e validação de novas configurações. [9] https://networkx.github.io/ [10] http://neo4j.com/

computadores com processador Intel Core I7-4790 3.6GHz x8 e com 16GB RAM. Por outro lado, depois de processados os dados de entrada, o NetworkX e o Neo4j permitiram executar as consultas de forma simples e rápida.

Através dos grafos foi possível aplicar análises mais aprofundadas sobre a dinâmica do ecossistema de PTTs na Internet do Brasil, cujos resultados são apresentados na próxima seção com a descrição das métricas. Os grafos individuais de todos os PTTs, suas ferramentas complementares (algoritmos) e as análises detalhadas fazem parte dos arquivos do *dataset* que estamos disponibilizando publicamente.

4. Análise e Discussão dos Resultados

Nesta seção discutimos os resultados obtidos a partir das análises dos dados coletados, compreendendo os seguintes estudos: (*i*) classificação dos membros dos PTTs; (*ii*) grau dos vértices; (*iii*) profundidade isolada e acumulada; (*iv*) densidade de pares; (*v*) ocorrência de AS-Prepend; e (*vi*) comunidades k-clique. Por limitação de espaço apresentamos apenas os resultados de quatro PTTs sem filtros: DF, MG, RS e VIX.

Nos estudos tabelados optamos por inserir uma coluna denominada Brasil equivalente à média dos grafos de todos os PTTs com seus devidos valores de desvio padrão. O alto valor observado nos desvios padrões mostra que há diversidade no panorama nacional, sendo importante destacar que a diferença entre PTTs maiores e menores também influncia nessa dispersão. Por exemplo, em PTTs menores que possuem menos membros é normal que haja maior grau de conexão (densidade) porque há menos combinações possíveis, embora essa característica não possa ser assumida como regra.

4.1. Classificação dos Membros

Um primeiro esforço de organização dos trabalhos de análise foi classificar todos os SAs dos PTTs brasileiros, tarefa que nos permitiu identificar o perfil atual dos membros interessados em *peering* para troca de tráfego nas diversas regiões do Brasil. No *dataset* compartilhado pode ser encontrada a classificação individual de todos os PTTs, enquanto que na tabela 2 trazemos uma síntese com o resultado da classificação geral do cenário nacional e algumas classificações individuais, incluindo o PTT-SP.

Ao observar os resultados da tabela 2, tanto em nível Brasil como também individualmente por PTT, fica evidente que a maior parcela dos membros interessados em *peering* consiste em provedores de acesso de abrangência local. Esse resultado era esperado e pode ser explicado pelo interesse econômico dos provedores de acesso em trocar o máximo de tráfego possível através de acordos multilaterais na infraestrutura compartilhada dos PTTs, uma vez que dessa maneira há economia decorrente da não utilização do trânsito provido pelas grandes operadoras de telecomunicações de abrangência estadual ou nacional. A iniciativa de conexão dos provedores menores aos PTTs é importante porque traz impacto direto nos preços praticados aos clientes em um cenário local de competição entre vários provedores de acesso.

A única exceção ocorreu no PTT-DF em que a presença do governo é predominante em relação às demais categorias, uma particularidade regional da capital federal do Brasil que possui como membros de destaque: o Senado Federal, a Polícia Federal, o Serviço Federal de Processamento de Dados (Serpro), a Empresa de Tecnologia da Informação da Previdência Social (Dataprev), a Telebras, entre outros.

Tabela 2. Perfil de SAs nos PTTs do Brasil

Classificação	Brasil (*)	DF	MG	RS	SP	VIX
1. Provedor de Internet	65,1% ± 20%	37,5%	55,9%	68,0%	73,1%	75,0%
1.1 Provedor de Trânsito	8,6% ± 09%	20,8%	14,7%	5,0%	5,6%	10,0%
1.2 Provedor de Acesso	56,5% ± 21%	16,7%	41,2%	63,0%	67,5%	65,0%
2. Provedor de Serviços	10,1% ± 07%	8,3%	8,8%	5,0%	12,5%	5,0%
2.1 Provedor de Conteúdo	3,2% ± 06%	0,0%	2,9%	3,0%	4,7%	0,0%
2.2 Provedor de Hospedagem	6,8% ± 05%	8,3%	5,9%	2,0%	7,8%	5,0%
3. Organização Pública	12,3% ± 21%	37,5%	20,6%	11,0%	4,4%	15,0%
3.1 Universidade Pública	1,8% ± 19%	0,0%	0,0%	2,0%	1,1%	0,0%
3.2 Governo	8,8% ± 13%	33,3%	17,6%	8,0%	2,2%	15,0%
3.3 Outros	1,8% ± 03%	4,2%	2,9%	1,0%	1,1%	0,0%
4. Organização Privada	12,6% ± 09%	16,7%	14,7%	16,0%	10,0%	5,0%
4.1 Universidade Privada	0,7% ± 03%	0,0%	2,9%	4,0%	0,0%	0,0%
4.2 Empresa Privada	10,4% ± 09%	16,7%	8,8%	10,0%	8,9%	5,0%
4.3 Outros	1,5% ± 09%	0,0%	2,9%	2,0%	1,1%	0,0%

(*) Média de todos os 26 PTTs brasileiros.

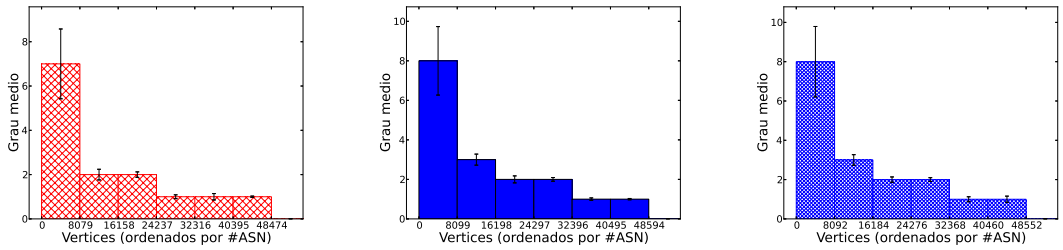
Foi observada baixa participação dos provedores de conteúdo nos PTTs em operação nas diversas regiões brasileiras, a exemplo de jornais, revistas, rádios, emissoras de televisão, etc; sendo que a maioria dos representantes da classe *provedor de conteúdo* são empresas que operam *Content Delivery/Distribution Networks* (CDN). Esse baixo resultado evidencia que ainda são poucos os provedores de conteúdo explorando os benefícios de desempenho do *peering* nos PTTs, principalmente no que diz respeito à menor latência ao diminuir o diâmetro da comunicação fim-a-fim. Por outro lado, reconhecemos como explicação plausível desse número o interesse dos provedores de conteúdo na contratação das grandes empresas de CDN para distribuição do seu conteúdo nas proximidades dos seus usuários (clientes) no panorama de um vasto território geográfico, em contraste aos PTTs que trazem benefícios apenas no panorama local.

Também é possível notar a baixa presença de empresas privadas nos PTTs, o que pode ser explicado pela motivação de redundância das empresas em implementar conexões *multi-homed* através de duas ou mais operadoras de telecomunicações que tenham alcance a toda a Internet, em contraste aos PTTs que, normalmente, têm alcance limitado à sua região geográfica.

Foi constatado que 97,72% dos SAs brasileiros conectados nos PTTs optaram por *peering* aberto com acordo de tráfego multilateral (ATM), um número bastante alto que evidencia os esforços da política pública do NIC.br no projeto PTTMetro. Apenas 2,28% optou por *peering* privado através de acordos de tráfego bilateral (ATB), sendo que praticamente a totalidade dessa porcentagem são provedores de trânsito, a destacar: Brasil Telecom, Embratel, Global Crossing, Level3, NTT, Oi, Telefonica e TIM. Essa observação é coerente porque as grandes operadoras de telecomunicações que vendem trânsito para provedores de acesso locais e regionais não têm motivação econômica para trocar tráfego abertamente com qualquer outro SA que não seja de natureza similar, uma vez que provedores menores são potenciais clientes interessados na compra de trânsito. Como exceção, as seguintes grandes operadoras optaram por ATM: Algar Telecom, Brasil Telecom (em Brasília), Cemig Telecom, Claro, CTBC, GVT, Internexa (Colombiana), NET, Nextel, Oi (em BH, Londrina, Manaus e Natal), Sercomtel e Vivo.

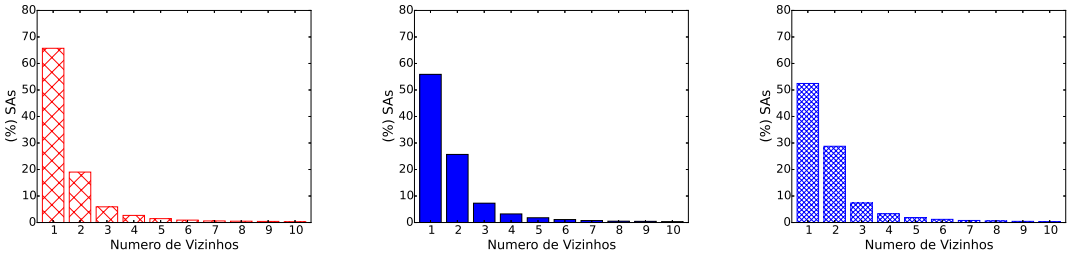
4.2. Graus dos Vértices

As figuras 2 e 3 mostram os graus dos vértices dos grafos referentes aos PTTs: DF, MG e VIX. Como a quantidade de vértices é muito grande para ser representada individualmente, optamos por ordenar o eixo x de maneira crescente por número de SA nos gráficos da figura 2. Dessa maneira é possível fazer uma leitura do grau médio dos vértices que equivalem a uma faixa específica de SAs, manobra que evidenciou que os SAs registrados há mais tempo (com números menores) têm maior grau médio, um resultado coerente se considerarmos que os vértices com maior grau são de operadoras de telecomunicações que têm mais adjacências pela natureza da atividade de venda de trânsito para provedores de acesso. Cada barra que representa o grau médio de uma faixa de SAs está acompanhada de um indicador do intervalo de confiança. Além dos graus médios, a figura 3 traz a distribuição dos valores exatos de graus dos nós.



(a) PTT-DF (b) PTT-MG (c) PTT-VIX

Figura 2. Grau Médio dos Grafos por Números de SAs



(a) PTT-DF (b) PTT-MG (c) PTT-VIX

Figura 3. Distribuição dos Graus dos Grafos

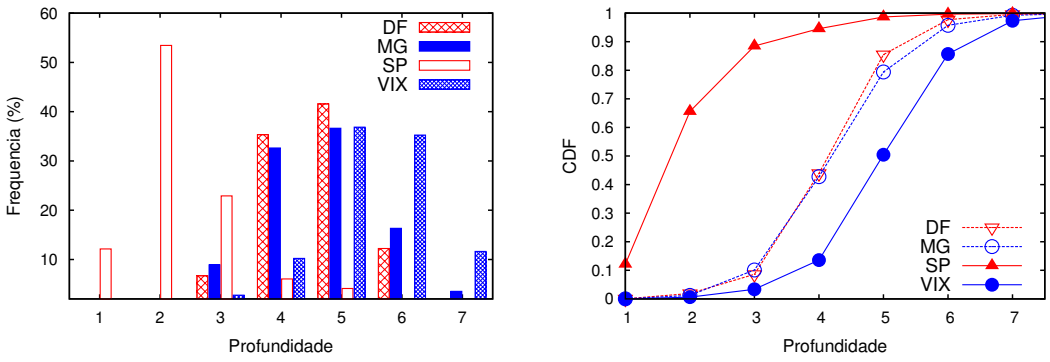
4.3. Profundidade/Diâmetro

O gráfico (a) da figura 4 traz um estudo da profundidade/diâmetro dos anúncios observados na tabela BGP dos PTTs com base no atributo AS-PATH. Ao observar os números de SAs que compõem o AS-PATH é possível determinar a quantidade de saltos e quais SAs devem ser atravessados por um pacote até que a origem anunciante de um determinado prefixo seja alcançada por um membro do PTT. Em relação a essa análise destacamos que identificamos e removemos informações duplicadas que poderiam comprometer os resultados, principalmente decorrentes da aplicação da política de AS-Prepend para fins de engenharia de tráfego. Um destaque em relação aos gráficos da figura 4 é que a profundidade igual a 1 indica que o AS-PATH é composto de apenas um SA, o que deve ser interpretado como SAs diretamente conectados no PTT que estão efetivamente fazendo

anúncio dos seus próprios prefixos (baixa porcentagem). As demais rotas com profundidade maior que 1 estão sendo aprendidas pelos membros dos PTTs a partir de outros SAs, ou seja, não estão sendo diretamente anunciadas pelos seus membros adjacentes.

Tomando o PTT-VIX no gráfico (a) como referência, por exemplo, lê-se que aproximadamente 40% (eixo y) do total de rotas anunciadas na tabela BGP possuem 5 SAs e outros 40% possuem 6 SAs no atributo AS-PATH, ou seja, cerca de 80% das rotas têm profundidade de ordem 5 ou 6. Ao observar os PTTs de forma conjunta, tem-se que a profundidade média das rotas anunciadas nos PTTs varia entre 4 e 6, sendo que há maior concentração de rotas com profundidade 5. Outro detalhe relevante é que a maior profundidade nos gráficos é de ordem 7, ou seja, as origens mais distantes dos anúncios que chegam nos PTTs estão até 7 saltos de distância, sendo que cada salto representa um SA (e não um roteador). No entanto, é importante destacar que nas tabelas BGP dos PTTs existem prefixos com profundidade maior que 7 sendo anunciados, mas a quantidade desses prefixos é mínima em relação à totalidade de anúncios. Esse fato fica evidente no gráfico (b) da figura 4 que apresenta uma contagem de saltos de forma cumulativa.

Os resultados das análises sobre o grafo do PTT-SP e do PTT-PR são discrepantes e devem ser desconsiderados porque os dados extraídos a partir de acesso público à tabela BGP (via LG) sofreram uma série de filtros que comprometeram essa massa de dados. Optamos por incluir o PTT-SP na apresentação dos resultados justamente para registrar esse alerta como um dos problemas que enfrentamos durante os trabalhos de análise e para evidenciar as implicações negativas de filtros em pesquisas dessa natureza.



(a) Profundidade Isolada (b) Profundidade Acumulada

Figura 4. Profundidade dos AS-PATH

4.4. Densidade de *Peering*

Além do potencial natural de ampliação da troca de tráfego no ecossistema de PTTs através da participação futura de novos membros, o estudo da densidade de pares correspondente à tabela 3 evidencia que há grande dispersão nas diferentes regiões de operação dos PTTs, com potencial real de aumento na troca de tráfego apenas entre os membros já existentes. A densidade de pares consiste na razão entre a quantidade de conexões (peering) existentes em um PTT com sua combinação de pares possíveis.

4.5. Engenharia de Tráfego Baseada em AS-Prepend

À medida em que os prefixos anunciados atravessam domínios BGP, o número do SA anterior é inserido à lista de SAs que foram percorridos, com o objetivo de descartar

Tabela 3. Densidade de Pares nos PTTs

Descrição da Métrica	Brasil (*)	DF	MG	RS	VIX
Peers Existentes (Conexões)	126	57	79	1.952	71
Combinação de *Peers* Possíveis	285	276	231	3.081	153
Densidade (%)	44,2% ± 23%	20,7%	34,2%	63,4%	46,4%

(*) Média dos 24 PTTs brasileiros sem filtro, ou seja, excluindo PTT-PR e PTT-SP.

conteúdo que tenha seu próprio SA no AS-PATH para evitar *loops*. No comportamento padrão do BGP as menores listas de AS-PATH serão preferíveis, ou seja, aqueles caminhos com menor profundidade/diâmetro, o que torna o atributo AS-PATH importante para selecionar a melhor rota quando existem múltiplas rotas para um mesmo destino. O atributo AS-PATH é frequentemente manipulado por um SA para adicionar (*prepend*) mais de uma vez seu próprio número de SA e, portanto, tornar menos atrativa a alcançabilidade de um prefixo através dele mesmo.

Os números da tabela 4 reforçam que AS-Prepend é frequentemente praticado na Internet, conforme pode ser observado nos estudos da quantidade total de rotas que sofreram AS-Prepend nos *Paths BGP* dos PTTs e também de SAs que praticam essa técnica para fins de políticas de roteamento. Cabe observar que o segundo bloco da tabela diz respeito a todos os SAs na Internet que foram observados a partir dos grafos dos PTTs, sejam eles membros ou não. No terceiro bloco da tabela são trazidos os números referentes apenas aos membros diretamente conectados nos PTTs, sendo importante observar que nem todos os membros de um PTT fazem anúncios.

Uma observação relevante é que o número de rotas nas tabelas BGP é um conceito diferente do número de prefixos. Sabe-se que atualmente a tabela BGP completa possui pouco mais de 512.000 prefixos, no entanto é comum as tabelas BGP dos PTTs possuirem milhares de linhas porque podem ter múltiplas rotas apontando para um mesmo prefixo.

Tabela 4. Ocorrência de AS-Prepend na Visão dos PTTs

Descrição da Métrica	Brasil (*)	DF	MG	RS	VIX
Quantidade de Rotas	932.989	559.159	434.264	1.947.453	2.663.751
Quantidade de Rotas com AS-Prepend	295.909	127.184	245.129	1.710.070	623.965
AS-Prepend X Rotas (%)	30,8% ± 22%	22,7%	56,4%	87,8%	23,4%
Quantidade de SAs	43.333	47.176	46.939	48.351	47.474
Quantidade de SAs com AS-Prepend	7.305	6.206	8.629	10.803	9.124
AS-Prepend X SAs (%)	16,1% ± 4%	13,2%	18,4%	22,3%	19,2%
Quantidade de Membros (Anunciantes)	18	24	22	79	18
Quantidade de Membros c/ AS-Prepend	6	7	6	36	5
AS-Prepend X Membros (%)	22,5% ± 19%	29,2%	27,3%	45,6%	27,8%

(*) Média dos 24 PTTs brasileiros sem filtro, ou seja, excluindo PTT-PR e PTT-SP.

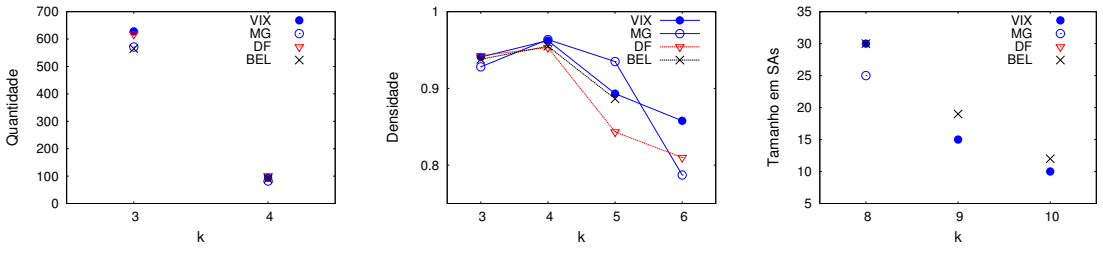
4.6. Comunidades k-Clique

Uma comunidade é composta por um conjunto de SAs (vértices), sejam eles membros diretamente conectados em um PTT ou mesmo externos. Os métodos de extração de comunidades da Internet que se destacam na literatura são *k-clique*, *k-dense* e *k-core* [Orsini et al. 2013]. A principal característica em comum desses métodos é detectar conjuntos de SAs densos no que diz respeito à conectividade.

O método de comunidades k-clique foi escolhido porque possui característica peculiar de fornecer comunidades sobrepostas [Palla et al. 2005], o que permite separar comunidades que tenham certa quantidade de nós em comum. Um k-clique é um subgrafo completo com k nós. Dois k-cliques são adjacentes se possuirem $k-1$ nós em comum. Uma comunidade k-clique, ou *comunidade(k)*, é o conjunto maximal formado pela união de todos os k-cliques adjacentes. Outra propriedade é que em cada comunidade k-clique existe uma e apenas uma comunidade (k-1)-clique tal que *comunidade(k)* é um subgrafo de *comunidade(k-1)* [Gregori et al. 2011], ou seja, com menores valores de k é possível identificar mais comunidades do que seria identificado com maiores valores de k.

Através desse estudo foi possível encontrar comunidades e caracterizar seus membros de maneira a ultrapassar a perspectiva localizada dos PTTs, fornecendo uma visão da Internet propriamente dita. Para valores menores de k, principalmente 3 ou 4, tendo como referência a visão dos PTTs estudados, a quantidade de comunidades encontradas na Internet é similar (figura 5(a)), sendo que essas comunidades possuem alta densidade de conexão entre seus membros (figura 5(b)). Igualmente à densidade previamente apresentada no contexto específico dos PTTs (sub-seção 4.4), definimos a densidade desse estudo como a média das densidades individuais das comunidades k-clique.

Considerando a caracterização previamente apresentada na sub-seção 4.1, uma observação interessante no contexto do ecossistema dos PTTs foi a identificação de comunidades k_{max}, cujos membros têm perfil similar. Definimos uma comunidade k_{max} como um conjunto não vazio de SAs que forma uma comunidade k-clique com o valor máximo de k. Existem poucas comunidades k_{max} que possuem SAs com identificadores (ASN) de 3 ou 4 dígitos em sua grande maioria. Tais SAs são predominantemente provedores de trânsito *tier-1*, com poucos casos de presença de grandes empresas como Facebook, Microsoft, Amazon, etc. Em comunidades com menores valores de k existem mais SAs provedores de acesso (com ASNs de 5 dígitos). A figura 5(c) mostra a quantidade de SAs das maiores comunidades encontradas, além de permitir inferir os valores de k_{max}, observando que o PTT-DF não aparece no gráfico porque seu k_{max} é apenas 6.



(a) Comunidades k3 e k4 (b) Densidade (c) Maiores Comunidades

Figura 5. Comunidades k-Clique

5. Limitações

Igualmente importante ao esforço de análise apresentado nesse artigo é o reconhecimento de algumas limitações de pesquisas intencionadas em construir uma topologia da Internet em nível de SAs [Haddadi and Bonaventure 2013]. Embora tenhamos confiança nos resultados da nossa metodologia de análise do *dataset* que estamos compartilhando, é importante destacar que as informações extraídas dos servidores públicos não representam

a totalidade da troca de tráfego que realmente ocorre entre os diversos membros dos SAs brasileiros, mas apenas uma fração equivalente a tudo aquilo que pode ser publicamente observado. Em termos de *peering* nos PTTs, somente os acordos de tráfego multilaterais (ATM) são espelhados nos servidores públicos e podem ser visualizados a partir dos LGs, sendo que os acordos de tráfego bilaterais (ATB) não podem ser visualizados porque são diretamente estabelecidos entre dois SAs de maneira privada.

Outro detalhe relevante é que os LGs públicos deveriam espelhar todas as rotas presentes no(s) RS(s) dos PTTs, no entanto é impossível garantir que as observações a partir do acesso público não tenham sofrido filtragem prévia por alguma política administrativa na gestão dos PTTs. Por exemplo, durante nossas análises ficou evidente que as tabelas BGP extraídas do PTT-SP e do PTT-PR não estavam completas ao compará-las com as tabelas dos demais PTTs, informação que foi confirmada pelo NIC.br com a justificativa de problemas de escalabilidade. Essa limitação explica a maior concentração de rotas com menor profundidade e a baixa porcentagem de densidade resultante da análise do PTT-SP, enquanto que nos demais PTTs há maior concentração de rotas com maior profundidade e maior porcentagem de densidade.

Uma das informações constantes no *dataset* que compartilhamos são os resumos das *communities* BGP configuradas nos PTTs, no entanto não pudemos utilizar esses dados para inferir *links* ocultos seguindo a metodologia proposta em [Giotsas et al. 2013] porque fomos informados pelo NIC.br de que não existe um sistema de codificação padronizado para utilização das *communities* no PTTMetro brasileiro.

6. Trabalhos Futuros

Uma primeira extensão importante da nossa metodologia é a construção de uma análise temporal a partir de amostras de dados de períodos futuros, fornecendo diferentes *snapshots* que permitam compreender a evolução da dinâmica do ecossistema de PTTs no Brasil e construir um modelo de predição. Também pretendemos expandir nosso estudo no sentido de realizar uma análise dos tamanhos dos blocos de prefixos IPv4 e IPv6 constantes nas tabelas BGP, o que permitirá comparar os dois protocolos e verificar qual perfil de membro tende a anunciar qual tamanho de bloco.

Estamos trabalhando em análises mais aprofundadas sobre as comunidades k-clique porque enxergamos a possibilidade de inferir novos resultados, principalmente no sentido de caracterização dos SAs e de compreensão da Internet. Pretendemos expandir nossa visão das comunidades na Internet a partir da extração de dados de servidores de rotas disponibilizados publicamente pelas grandes operadoras de telecomunicações. Esse conjunto de dados será utilizado para subsidiar pesquisas relacionadas a *SDN Peering* que estamos desenvolvendo na perspectiva de que a combinação de SDN com PTTs viabilize novas aplicações que não são possíveis atualmente [Gupta et al. 2014], como por exemplo o *peering* baseado em conteúdo ou aplicação e novas maneiras de aplicar políticas entre domínios para fins de engenharia de tráfego.

7. Conclusão

Este artigo apresentou um estudo que acreditamos ter sido o primeiro esforço de compreensão do ecossistema de PTTs públicos em operação no Brasil, compilando informações

coletadas a partir de servidores públicos que nos permitiram reproduzir os grafos de conectividade dos PTTs brasileiros para compreensão da sua anatomia, oportunidade em que identificamos diversos aspectos da estrutura desse ecossistema.

Agradecimentos

Agradecemos ao Centro de Inovação da Ericsson Telecomunicações S/A do Brasil pelo suporte financeiro. Também ao NIC.br pela colaboração, especialmente aos colegas Antonio Moreiras (CEPTRO.br) e Antonio Galvão de Rezende Filho (PTT.br).

Referências

Ager, B., Chatzis, N., Feldmann, A., Sarrar, N., Uhlig, S., and Willinger, W. (2012). *Anatomy of a Large European IXP*. SIGCOMM'12, August 13-17, Helsinki, Filand.

Chatzis, N., Smaradgakis, G., Feldmann, A., and Willinger, W. (2013). *There is More to IXPs than Meets the Eye*. CCR ACM SIGCOMM. Volume 43, Number 5. October.

Chen, K. and et al (2009). *Where the Sidewalk Ends, Extending the Internet AS Graph Using Traceroutes from P2P Users*. CoNEXT'09, December 01-04, Rome, Italy.

Giotsas, V., Luckie, M., Huffaker, B., and Claffy, K. (2014). *Inferring Complex AS Relationships*. IMC'14. November 6-7. Vancouver, BC, Canada.

Giotsas, V., Zhou, S., Luckie, M., and Claffy, K. (2013). *Inferring Multilateral Peering*. CoNEXT'13, December 9-12. Santa Barbara, California, USA.

Gregori, E., Lenzini, L., and Orsini, C. (2011). *k-clique Communities in the Internet AS-level Topology Graph*. 31st ICDCSW. IEEE.

Gupta, A., Vanbever, L., Shabaz, M., Donovan, S. P., Schlinker, B., Feamster, N., et al. (2014). *SDX - A Software Defined Internet Exchange*. SIGCOMM'14. Chicago, USA.

Haddadi, H. and Bonaventure, O. (2013). *Recent Advances in Networking. Chapter 1: Internet Topology Research Redux*. ACM SIGCOMM eBook. Volume 1. August.

Labovitz, C., Iekel-Johson, S., McPherson, D., Oberheide, J., and Jahanian, F. (2010). *Internet Inter-Domain Traffic*. SIGCOMM'10, Aug 30 - Sep 3, New Delhi, India.

Lodhi, A., Larson, N., Dhamdhere, A., Dovrolis, C., and Claffy, K. (2014). *Using PeeringDB to Understand the Peering Ecossystem*. CCR ACM SIGCOMM. April.

Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., and Claffy, K. (2013). *AS Relationships, Customer Cones, and Validation*. IMC'13. Barcelona, Spain.

Norton, W. B. (2014). *The Internet Peering Playbook: Connecting to the Core of the Internet*. DrPeering Press. 2014 Edition. USA.

Orsini, C., Gregori, E., Lenzini, L., and Krioukov, D. (2013). *Evolution of the Internet k-Dense Structure*. IEEE/ACM Transactions on Networking. September 16.

Palla et al. (2005). *Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society*. Nature. Volume 435. Number 7043.

Restrepo, J. C. C. and Stanojevic, R. (2012). *A History of an Internet eXchange Point*. CCR ACM SIGCOMM. Volume 42, Number 2, April.

Schlinker, B., Zarifis, K., Cunha, I., Feamster, N., and Katz-Bassett, E. (2014). *PEERING: An AS for US*. HotNets'14. October 27-28. Los Angeles, California, USA.

Impact: Um detector de falhas baseado na relevância dos processos e no grau de confiança no sistema

Anubis Graciela de Moraes Rossetto[1], Luciana Arantes[2], Pierre Sens[2], Cláudio F. R. Geyer[1]

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil,

[2]Sorbonne Universités, UPMC Université. Paris 06, CNRS, Inria, LIP6

{agmrossetto, geyer}@inf.ufrgs.br

{luciana.arantes, pierre.sens}@lip6.fr

Abstract. *This work proposes a new and flexible unreliable failure detector, denoted the Impact Failure Detector (FD), whose output is related to the trust level of a set of processes. By expressing the relevance of each node by an impact factor value as well as a margin of acceptable failures of the system, the Impact FD enables the user to tune the failure detection configuration in accordance with the requirements of the application: in some scenarios, the failure of low impact or redundant nodes does not jeopardize the confidence in the system, while the crash of a high impact factor one may seriously affect it. A softer or stricter monitoring is thus possible. Performance evaluation results using real PlanetLab [PlanetLab 2014] traces confirm the degree of flexibility of our failure detector and, due to the margin of failure, the number of false responses may be reduced when compared to traditional unreliable failure detectors.*

Resumo. *Este trabalho propõe um novo detector de falhas não confiável chamado Impact Failure Detector (FD), cuja saída está relacionada ao nível de confiança em um conjunto de processos. Ao expressar a relevância de cada processo por um valor de fator de impacto, bem como por uma margem de falhas aceitáveis do sistema, o Impact FD permite ao usuário ajustar a configuração do detector de falhas de acordo com os requisitos da aplicação: em certos cenários, a falha de um processo de baixo impacto ou redundante não compromete a confiança no sistema, enquanto a falha de um processo de alto fator de impacto pode afetá-la seriamente. Assim, é possível um monitoramento com maior ou menor rigor. Os resultados da avaliação de desempenho usando traces reais do PlanetLab [PlanetLab 2014] confirmam o grau de flexibilidade do nosso detector de falhas e, devido à margem de falhas, o número de falsas respostas pode ser reduzido quando comparado a detectores de falhas tradicionais.*

1. Introdução

Um detector de falhas não confiável pode ser visto como um óraculo que fornece informações, nem sempre corretas, sobre falhas de processos. A maioria dos detectores é baseada no modelo binário no qual os processos monitorados são ou "trusted" ou "suspected". Assim, muitos dos detectores de falhas existentes, como os definidos em [Chandra and Toueg 1996] e [Bertier et al. 2003], fornecem como saída o conjunto de

processos que atualmente são suspeitos de terem falhado (*crashed*). Uma abordagem não binária está presente em [Hayashibara et al. 2004], o detector de falhas Accrual, que oferece como saída o nível de suspeita de cada processo em uma escala contínua.

Neste artigo apresentamos um novo detector de falhas não confiável, o Impact Failure Detector (Impact FD) que fornece como saída um nível de confiança em relação a um conjunto S de processos monitorados. A saída pode ser considerada como o grau de confiança em S, ou seja, a confiança no conjunto de processos como um todo. Para este fim, um fator de impacto (*impact factor*) é atribuído a cada processo de S e um parâmetro limiar (*threshold*) define um valor limite, sobre o qual, o grau de confiança em S não é afetado. O fator de impacto indica a importância relativa do processo no conjunto S, enquanto o *threshold* oferece uma margem de flexibilidade para falhas e falsas suspeitas, permitindo assim uma maior tolerância à instabilidade do sistema. Por exemplo, em uma rede instável pode haver muitas falsas suspeitas ou mesmo falhas, mas, devido ao valor do *threshold*, o sistema atende ao grau de confiabilidade. No caso do Impact FD indicar que a confiabilidade em S está abaixo do limite definido pelo *threshold*, o usuário pode decidir qual medida tomar (urgente ou não, em relação ao nível de confiança indicado pelo Impact FD). Além disso, no conceito do Impact FD, os processos monitorados podem ser agrupados com base em algum critério como, por exemplo, tipo dos processos (nodos, sensores) ou relevância destes. Observe que, juntamente com o *threshold*, tal abordagem de grupo pode caracterizar redundância de processos.

As características e flexibilidade do Impact FD permitem que ele atenda a diferentes necessidades, podendo, desta forma, ser aplicado a diversos cenários distribuídos. Por exemplo, em redes de sensores sem fio (RSSFs) ubíquas, frequentemente utilizadas para monitorar as condições físicas de regiões geográficas, onde os sensores são de diferentes tipos (de controle de humidade, de temperatura, etc.) e o número de sensores distribuídos na região varia segundo estes tipos. Sensores são sujeitos a falhas e a redundância, neste caso, garante a cobertura da região e a conectividade da rede. Assim, poderíamos considerar a região como um único conjunto cujos sensores do mesmo tipo são agrupados em subconjuntos. Esta abordagem permite a definição de um *threshold* que é igual ao número mínimo de sensores que cada subconjunto deve ter, a fim de manter a conectividade e a aplicação funcionando. Além disso, em algumas situações pode haver a necessidade de reconfigurar dinamicamente o grau de redundância. Neste caso, basta mudar o valor do *threshold*. Um outro exemplo seria um sistema com um servidor principal que oferece um determinado serviço com uma certa qualidade de serviço X (vazão, tempo de resposta, etc.). Se este falhar, N servidores backup tem capacidade para substituí-lo, visto que cada backup oferece o mesmo serviço mas com uma qualidade de serviço X/N. Neste cenário, tanto o fator de impacto do servidor principal quanto o *threshold* teriam o valor de $N * Iback$, sendo $Iback$, o valor de impacto dos servidores backups, ou seja, o sistema não é confiável no caso em que o servidor principal e um ou mais dos N servidores falharem (ou suspeitos de estarem falhos).

Vale ressaltar que como a saída fornecida pelo Impact FD não está vinculada ao identificador (ID) dos processos, ele também pode ser aplicado a um contexto em que a composição (*membership*) do sistema não é conhecida ou mesmo em sistemas anônimos (processos não possuem ID) ou homônimos (processos podem ter o mesmo ID).

O restante deste artigo está organizado nas seguintes seções: A Seção 2 apresenta

alguns conceitos básicos sobre detectores de falhas não confiáveis. O Impact FD e alguns exemplos de suas classes são descritos na Seção 3. A Seção 4 resume alguns resultados de avaliação preliminares conduzidos com *traces* reais do PlanetLab [PlanetLab 2014]. A Seção 5 discute alguns trabalhos relacionados. A Seção 6 apresenta a conclusão e os trabalhos futuros.

2. Detector de falhas não confiável

Uma importante abstração para o desenvolvimento de sistemas distribuídos tolerantes a falhas é o detector de falhas não confiável [Chandra and Toueg 1996]. Ele tem por objetivo encapsular a incerteza do atraso de comunicação entre duas entidades distribuídas. Neste trabalho considera-se que existe um processo por nodo (site). Além disso, o termo *processo* pode significar um nodo, um sensor ou um site. Definimos como *correto*, um processo que nunca falha durante toda a execução.

Detectores de falhas não confiáveis são caracterizados por duas propriedades, *completude* (*completeness*) e *acurácia* (*acuracy*), tal como definido em [Chandra and Toueg 1996]. A completude está relacionada com a capacidade do detector suspeitar permanentemente de falhas dos processos, enquanto a acurácia diz respeito à capacidade de não suspeitar de processos corretos. No mesmo artigo, os autores classificam detectores de falhas de acordo com duas propriedades de completude e quatro propriedades de acurácia, as quais combinadas geram oito classes de detectores de falhas.

2.1. Implementação de detectores de falhas

A literatura contém várias propostas de implementação de detectores de falhas não confiáveis que normalmente exploram uma abordagem baseada em tempo (*timer-based*) ou em padrão de mensagem (*message-pattern*).

Na abordagem baseada em tempo, as implementações fazem uso de temporizadores para detectar falhas em processos. Todo processo q envia periodicamente uma mensagem de controle (*heartbeat*) ao processo p responsável pelo monitoramento de q. Se p não receber tal mensagem de q após a expiração de um temporizador, ele acrescenta q à sua lista de processos suspeitos. O uso dos *timeouts* (limite máximo de tempo) pressupõe que o sistema seja síncrono ou, após um tempo, se comporte de maneira síncrona (sincronia parcial) [Chandra and Toueg 1996]. Já a abordagem baseada em padrão de mensagem não usa qualquer mecanismo de tempo limite de espera por uma resposta. [Mostefaoui et al. 2003] propõem uma implementação que faz uso de um mecanismo de requisição-resposta (*query-response*). Um processo p transmite uma mensagem $QUERY$ para os n nodos que ele monitora e então espera as respostas ($RESPONSE$) correspondentes de α processos ($\alpha \leq n$, tradicionalmente, $\alpha = n - f$, onde f é o número máximo de falhas). Uma consulta emitida por p termina quando recebeu α respostas. Os demais processos, cujas respostas não foram consideradas, são vistos como suspeitos de terem falhado. Um processo transmite mensagens $QUERY$ repetidamente se não falhar. Se, na próxima requisição-resposta, p recebe uma resposta de um processo suspeito q, então p remove q de sua lista de suspeitos. Esta abordagem considera uma ordem relativa para a recepção de mensagens em que sempre, ou após um tempo, alguns nodos se comunicam mais rapidamente que outros.

3. Detector de Falhas Impact

Consideramos um sistema distribuído que consiste de um conjunto[1] finito de processos $\Pi = \{q_1, \ldots, q_n\}$, com $|\Pi| = n$. As falhas são apenas por parada (*crash*). Outros tipos de falhas (por exemplo *misbehavior*, transitórias, etc) serão objeto de um trabalho futuro. Assumimos a existência de algum tempo global denotado T. Um padrão de falha é uma função $F : T \rightarrow 2^{\Pi}$, onde $F(t)$ é o conjunto de processos que falharam antes ou no tempo t. A função $correct(F)$ denota o conjunto de processos corretos, ou seja, aqueles que nunca pertencem ao padrão de falha (F), enquanto que $faulty(F)$ denota o conjunto de processos falhos, ou seja, o complemento de $correct(F)$ em relação a Π.

O Impact FD pode ser definido como um detector de falhas não confiável que fornece uma saída relacionada ao nível de confiança com relação a um conjunto de processos. Se o nível de confiança (*trust level*), fornecido pelo detector, é igual ou maior do que um determinado limiar (*threshold*), definido pelo usuário, a confiança no conjunto de processos é assegurada. Portanto, podemos dizer que o sistema é confiável.

Denotamos ${I_p}^S$ o módulo Impact FD do processo p e S é um conjunto de processos de Π. Cada processo $q \in S$ tem um fator de impacto (*impact factor*) associado ($I_q | I_q > 0 : I_q \in \mathbb{R}$). Além disso, o conjunto S pode ser particionado em m subconjuntos disjuntos. Observe que este recurso de agrupamento permite que S seja particionado de acordo com algum critério. Por exemplo, em um cenário em que há diferentes tipos de sensores, aqueles do mesmo tipo podem ser agrupados no mesmo subconjunto. Definimos então $S^* = \{S_1^*, S_2^*, ...S_m^*\}$ como o conjunto S particionado em m disjuntos subconjuntos, onde cada S_i^* é um conjunto composto de tuplas $\langle id, impact \rangle$, sendo id o identificador do processo e $impact$ o valor do fator de impacto do processo.

Quando invocado em p, o Impact FD (${I_p}^S$) retorna o valor ${trust\_level_p}^S$. O ${trust\_level_p}^S$ é um conjunto que contém o nível de confiança (*trust level*) de cada subconjunto S^*, expressando a confiança que p tem no conjunto S. Denotamos ${trusted_p}^S(t) = \{trusted_1(t), \ldots, trusted_m(t)\}$, onde cada $trusted_i(t)$ $(1 \leq i \leq m)$ contém os processos do subconjunto S_i^* que não são considerados falhos por p no tempo $t \in T$. Similarmente ao S_i^*, cada $trusted_i$ é composto da tupla $\langle id, impact \rangle$. A confiança (*trust level*) a $t \in T$ do processo $p \notin F(t)$ com relação ao conjunto S é a função ${trust\_level_p}^S$ tal que ${trust\_level_p}^S(t) = \{trust\_level_i(t) | trust\_level_i(t) = sum(trusted_i(t)); 1 \leq i \leq m\}$. A função $sum(set)$ retorna a soma do fator de impacto de todos os elementos de set.

Um limiar aceitável de falhas, denotado ${threshold}^S > 0$, caracteriza o grau aceitável de flexibilidade de falha em relação ao conjunto S^*. O ${threshold}^S$ está relacionado com o nível mínimo de confiança requerido para cada subconjunto, ou seja, é definido como um conjunto que contém o respectivo *threshold* de cada um dos m subconjuntos de S^*: ${threshold}^S = \{threshold_1, \ldots, threshold_m\}$.

O ${threshold}^S$ é usado pela aplicação para verificar a confiança nos processos de S. Se, para cada um dos m subconjuntos de S^*, o $trust\_level_i(t) \geq threshold_i$, S é considerado confiável (*trusted*) no tempo t por p, ou seja, a confiança de p em S não foi comprometida; caso contrário, S é considerado não confiável (*not trusted*).

[1]Neste trabalho, "conjunto" e "multiconjunto" (um elemento de um multiconjunto pode aparecer mais de uma vez) são usados indistintamente.

$S^* = \{\{\langle q_1, 1\rangle, \langle q_2, 1\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_4, 4\rangle, \langle q_5, 4\rangle, \langle q_6, 4\rangle\}\}$

t	F(t)	$trusted_p^S(t)$	$trust\_level_p^S(t)$	Status
1	$\{\{\langle q_2, 1\rangle\}, \{\}, \{\}\}$	$\{\{\langle q_1, 1\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_4, 4\rangle, \langle q_5, 4\rangle, \langle q_6, 4\rangle\}\}$	{1, 3, 12}	TRUSTED
2	$\{\{\langle q_2, 1\rangle\}, \{\}, \{\langle q_6, 4\rangle\}\}$	$\{\{\langle q_1, 1\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_4, 4\rangle, \langle q_5, 4\rangle\}\}$	{1, 3, 8}	TRUSTED
3	$\{\{\langle q_2, 1\rangle\}, \{\}, \{\langle q_5, 4\rangle, \langle q_6, 4\rangle\}\}$	$\{\{\langle q_1, 1\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_4, 4\rangle\}\}$	{1, 3, 4}	NOT TRUSTED

$threshold^S = \{1, 3, 8\}$

Figura 1. Exemplo da saída do detector de falhas

Dois pontos merecem ser ressaltados: (1) ambos o $impactfactor$ e o $threshold^S$ tornam a estimativa da confiança em S flexível. Por exemplo, pode acontecer que alguns processos em S a $t \in T$ estejam falhos ou suspeitos de falhos, mas S ainda é considerado confiável por p a t; (2) o $threshold^S$ aumenta a tolerância de S a falsas suspeitas.

Também é interessante notar que o Impact FD é facilmente configurável de acordo com as necessidades do ambiente. O $threshold^S$ pode ser ajustado de forma a fornecer um monitoramento com maior ou menor rigor. Essa capacidade de adaptação é essencial em ambientes dinâmicos, como por exemplo, na computação ubíqua. Observe também que o Impact FD pode ser empregado quando a aplicação precisa de informações sobre cada processo de S individualmente. Neste caso, cada processo deve ser definido como um subconjunto de S^*.

Na Figura 1, consideramos um conjunto S, onde S^* é composto por três subgrupos. Os valores do conjunto $threshold^S$ definem que os subconjuntos S_1 e S_2 devem ter um processo correto, pelo menos, e o subconjunto S_3 deve ter dois. Várias situações são mostradas e o conjunto S é considerado confiável quando, para cada subconjunto S_i^* $(1 \leq i \leq 3)$, o $trust\_level_i(t) \geq threshold_i$.

3.1. Propriedades

Como assinalado na seção anterior, a abordagem do Impact FD possibilita realizar o monitoramento do sistema com maior ou menor rigor. Diferentes classes de Impact FD podem ser definidas, mas que necessitam de diferentes propriedades. Com este objetivo, definimos as seguintes três propriedades.

A propriedade de *flexibilidade* denota a possibilidade do detector fornecer diferentes respostas que levam a um estado confiável sobre o conjunto de processos. Considere X como o conjunto que contém todos os possíveis subconjuntos de elementos que satisfazem o *threshold* definido:

$$X = TPowerSet(S^*, threshold) | TPowerSet(S^*, threshold) = \times\ PowerSet(S_i^*, threshold_i)$$

Inicialmente, a função *TPowerSet* gera o conjunto das partes[2] (ou Power Set) para cada subconjunto (S_i^*) de S^*. Após, são filtrados apenas os subconjuntos de S_i^* cuja a soma dos seus elementos é maior ou igual ao seu *threshold*. Por fim, é feito o produto cartesiano para gerar todas as possibilidades[3]. Assim, X contém todos os subconjuntos possíveis que atendem o $threshold^S$. Vejamos um exemplo:

[2]O conjunto das partes de qualquer S é o conjunto de todos os subconjuntos de S, incluindo o conjunto vazio e o próprio S.

[3]O $\times S_i^*$ é uma forma de abreviar o produto cartesiano quando existem vários conjuntos, por exemplo S_1^*, S_2^*, S_3^* $(X = (S_3^*) \times (S_2^*) \times (S_3^*))$

$S^* = \{\{\langle q_1, 2\rangle, \langle q_2, 4\rangle, \langle q_3, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}$
threshold^S = {*3, 3*}
$X = TPowerset(S^*, threshold^S)$
PowerSet(S_1^**,* $threshold_1$*)* = $\{\{\langle q_2, 4\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_1, 2\rangle, \langle q_2, 4\rangle\}, \{\langle q_1, 2\rangle, \langle q_3, 3\rangle\}, \{\langle q_2, 4\rangle\ \langle q_3, 3\rangle\},$
$\{\langle q_1, 2\rangle, \langle q_2, 4\rangle, \langle q_3, 3\rangle\}\}$
PowerSet(S_2^**,* $threshold_2$*)* = $\{\{\langle q_6, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle\}, \{\langle q_4, 1\rangle, \langle q_6, 3\rangle\}, \{\langle q_5, 2\rangle, \langle q_6, 3\rangle\},$
$\{\langle q_4, 1\rangle, \langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}$
X = PowerSet(S_1^**,* $threshold_1$*)* $\times$ *PowerSet(*S_2^**,* $threshold_2$*)*
X = $\{\{\{\langle q_2, 4\rangle\}, \{\langle q_6, 3\rangle\}\}, \{\{\langle q_2, 4\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle\}\}, \{\{\langle q_2, 4\rangle\}, \{\langle q_4, 1\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_2, 4\rangle\},$
$\{\langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_2, 4\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_3, 3\rangle\}, \{\langle q_6, 3\rangle\}\}, \{\{\langle q_3, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle\}\},$
$\{\{\langle q_3, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_3, 3\rangle\}, \{\langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_3, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle, \langle q_6, 3\rangle\}\},$
$\{\{\langle q_1, 2\rangle, \langle q_2, 4\rangle\}, \{\langle q_6, 3\rangle\}\}, \{\{\langle q_1, 2\rangle, \langle q_2, 4\rangle\}, \ldots\}$

Prop. 1 $PR(\diamondsuit IT)$ ***(Impact Threshold)***: Para um detector de falhas de um processo p, correto, existe um tempo após o qual o conjunto ${trusted_p}^S$ é um subconjunto de X.

$$PR(\diamondsuit IT) \equiv \exists t \in T, p \in correct(F), \forall t' \geq t, {trusted_p}^S(t') \subset X$$

Prop. 2 ${Impact\ strong\ completeness_p}^S$: Para um detector de falhas de um processo p, correto, existe um tempo após o qual p não confia em qualquer processo falho de S.

$$\exists t \in T, p \in correct(F), \forall q \in (faulty(F) \cap S), \forall t' \geq t, \langle q, \_\rangle \notin {trusted_p}^S(t')$$

Prop. 3 ${Eventual\ impact\ strong\ accuracy_p}^S$: Para um detector de falhas de um processo p, correto, existe um tempo após o qual todos os processos de ${trusted_p}^S$ são corretos.

$$\exists t \in T, \forall t' \geq t, p \in correct(F), \forall \langle q, \_\rangle \in {trusted_p}^S(t'), q \in (correct(F) \cap S)$$

3.2. Exemplos de algumas classes do Impact FD

Assim, quando uma ou mais das propriedades acima descritas são satisfeitas, podemos definir as seguintes classes do Impact FD:

$\diamondsuit$**IT** (*Eventual Impact Threshold Class*): esta classe pode ser implementada quando, considerando nossas definições para o *impact factor* e o *threshold*, o sistema subjacente satisfaz a Prop. 1 (PR ($\diamondsuit IT$)).

Definition 1 ($\diamondsuit IT$): Para um detector de falhas de um processo p, correto, existe um tempo após o qual o ${trust\_level_i}^S(t)$ é maior ou igual ao ${threshold_i}^S$, para todos os subconjuntos de S.

$$\exists t \in T, p \in correct(F), \forall t' \geq t, trust\_level_i(t') \geq threshold_i, \forall 1 \leq i \leq m$$

Na próxima subseção apresentamos uma implementação baseada em *query-response* para a classe $\diamondsuit IT$.

$\diamondsuit$**ICT** (*Eventual Correct Impact Class*): Para o processo p, o conjunto S e um *threshold* definido, as Prop. 1 e Prop. 2 são satisfeitas.

$\diamondsuit$**IPT** (*Eventual Perfect Impact Class*): Para o processo p e o conjunto S, as Prop. 1, Prop. 2 e Prop. 3 são satisfeitas.

3.3. Implementação de um FD da classe $\diamondsuit IT$

O conceito do Impact FD pode ter diferentes implementações de acordo com as premissas do sistema: modelo de sincronismo, o conhecimento ou não do processo p a respeito da composição de S (*membership*), o tipo dos nodos (identificável, anônimo, homônimo), etc. Apresentamos nesta seção um algoritmo que implementa a classe $\diamondsuit IT$ em um sistema assíncrono, baseado no modelo *query-response*. No entanto, vale ressaltar que outra implementação possível seria, por exemplo, com *heartbeat* e *timeout*, considerando um sistema parcialmente síncrono.

Na presente implementação considera-se que a composição do sistema (*membership*) e o número de processo de S não são conhecidos incialmente pelos processos. Além disso, todos os processos se comportam como um nodo de monitoramento, ou seja, $p \in S$. Assume-se que a rede é totalmente conectada, ou seja, $\Lambda = (p, q)|p, q \in \Pi$. Os processos podem invocar a primitava *broadcast(m)* para enviar uma mensagem m para todos os processos do sistema. Se p falha ao transmitir m, esta é recebida por um subconjunto arbitrário de processos. Os links são confiáveis, ou seja, links não perdem, não duplicam, não corrompem, nem geram espúria com relação às mensagens. A propriedade $PR(\diamondsuit IT)$ é satisfeita.

Algoritmo 1 descreve a implementação do $\diamondsuit IT$ FD no processo p em relação à S. Ele se executa por rodadas. A cada nova rodada, caso não falhe, um processo envia por difusão uma mensagem *QUERY*. O intervalo de tempo entre duas rodadas consecutivas é finito mas arbitrário. Quando o processo q recebe uma mensagem *QUERY* de um processo p, q lhe confirma a recepção com uma mensagem *RESP*.

O processo p recebe como entrada seu próprio fator de impacto (I_p), o conjunto a qual pertence ($subset_p$), o *threshold* de cada subconjunto de S^* (conjunto $threshold^S$), o número de subconjuntos de S^* (m), e o número máximo de mensagens que deve esperar (α). Este último é o conjunto $\alpha = \{\alpha_1, \ldots, \alpha_m\}$, onde cada α_i corresponde a um valor limite para o número de mensagens que deve esperar dos processos do subconjunto S_i^*. Por exempo, se f_i denota o número máximo falhas do subconjunto S_i^*, $\alpha_i \leq |S_i^*| - f_i$. No caso do subconjunto de p (que responde à sua própria mensagem *QUERY*), $\alpha_i \leq |S_i^*| - f_i - 1$. A seguinte notação é utilizada:

- r_p: contador de rodadas do processo p;
- *trusted*: é formado por $\{trusted_1, \ldots, trusted_m\}$, onde cada $trusted_i$ ($1 \leq i \leq m$) contém os processos de S_i não considerados falhos por p. Cada $trusted_i$ é um conjunto composto pela tupla $\langle id, impact \rangle$, onde id representa o identificador do processo e $impact$ o valor do fator de impacto do processo que pertence ao subset $trusted_i$;
- *local_known*: conhecimento atual que p tem sobre a composição do sistema (*membership*). É composto de subjconjuntos cujos elementos são do tipo $\langle id, impact \rangle$;
- *trust_level*: conjunto com o nível de confiança de cada subconjunto de processos;
- *X*: conjunto que compreende todos os subconjuntos possíveis formados pelos processos conhecidos por p, cuja soma dos seus elementos é maior ou igual ao *threshold* (veja Seção 3.1). Inicialmente é vazio pois a composição do sistema não é conhecida;
- *sum(set)* retorna a soma do fator de impacto de todos os elementos do conjunto/subconjunto set;
- *size(set)*: função que retorna o número de elementos do conjunto/subconjunto set;

- *Add(set, subset, $\langle q, impact \rangle$)*: função que insere o processo q no subconjunto $subset$ do conjunto set;
- *TPowerSet(set, threshold)*: função que retorna o conjunto que compreende todos os subconjuntos de set cuja a soma dos seus elementos é maior ou igual ao *threshold*.

Algoritmo 1 Algoritmo que implementa a classe $\diamondsuit IT$ FD no processo p

```
 1: Begin
    Input
 2:     I_p, subset_p, threshold^S, m, α

    Init
 3:     r_p ← 0
 4:     local_known ← ∅
 5:     trust_level ← ∅
 6:     X ← ∅
 7:     trusted ← ∅
 8:     c_trusted ← ∅
 9:     Add(local_known, subset_p, ⟨p, I_p⟩)
10:     Add(trusted, subset_p, ⟨p, I_p⟩)
11:     Add(c_trusted, subset_p, ⟨p, I_p⟩)

    Task T1
12:     loop
13:         broadcast(QUERY, r_p)
14:         wait until (size(trusted_i) ≥ α_i |1 ≤
    i ≤ m) or (trusted ⊆ X)
15:         c_trusted ← trusted
16:         trusted ← ∅
17:         Add(trusted, subset_p, ⟨p, I_p⟩)
18:         r_p ← r_p + 1
19:     end loop

    Task T2 - Response
20:     Upon reception of (RESP, r, I_q, subset_q)
    from q do
21:         if r = r_p then
22:             Add(trusted, subset_q, ⟨q, I_q⟩)
23:         end if
24:         if ⟨q, I_q⟩ ∉ susbset_q of local_known
    then
25:             Add(local_known, subset_q, ⟨q, I_q⟩)
26:             X ← TPowerSet(local_known, threshold^S)
27:         end if
28:     end

    Task T3 - Query
29:     Upon reception of (QUERY, r) from q
    do
30:         send(RESP, r, I_p, subset_p) to q
31:     end

    Task T4
32:     Upon invocation of Impact() do
33:         for i = 1 to m do        ▷ for each subset
34:             trust_level_i ← sum(c_trusted_i)
35:         end for
36:         return trust_level
37:     end
38: End
```

O algoritmo possui 4 tarefas. Na tarefa T1 de p há um laço infinito. Primeiro, a mensagem *(QUERY, r_p)* é enviada para todos os processos (linha 13). Em cada rodada (r_p), o processo p espera por pelo menos α_i respostas ($1 \leq i \leq m$) ou até que *trusted* seja um subconjunto de X (isto é, contém os processos que satisfazem o $threshold^S$) (linha 14). Por fim, o contador da rodada (r_p) é incrementado (linha 18).

A tarefa T2 trata a recepção das mensagens *(RESP, r, I_q $subset_q$)* enviadas por q. A mensagem contém a rodada, o fator de impacto e o subconjunto ao qual q pertence. Se a rodada r é igual a r_p, então $\langle q, I_q \rangle$ é adicionado ao conjunto $trusted_q$. Se q ainda não era conhecido por p, então $\langle q, I_q \rangle$ é adicionado a *local_known* no subconjunto $subset_q$ (linha 25). Neste caso, quando um novo processo é adicionado, X é atualizado (linha 26).

A tarefa T3 é responsável pela recepção da mensagem *QUERY*. Quando um processo p recebe uma mensagem *(QUERY, r)* do processo q (linha 29), p deve responder com uma mensagem *RESP* contendo a rodada, o seu fator de impacto e o número do seu subconjunto (linha 30).

A tarefa T4 trata a invocação da função Impact() (linha 32) que retorna o *trust_level* dos processos confiáveis dos subconjuntos de *trusted* (linha 36).

Neste artigo não apresentamos a prova de corretude do algoritmo em função do espaço limitado. Ela pode ser encontrada em [Rossetto et al. 2015].

4. Avaliação de desempenho

Nesta seção descrevemos inicialmente o ambiente no qual os experimentos foram realizados e as métricas de qualidade de serviço (QoS) utilizadas. Em seguida, apresentamos alguns resultados da avaliação com diferentes configurações de conjuntos de nodos no que diz respeito tanto ao fator de impacto quanto ao *threshold*, bem como a comparação com o detector de falhas proposto por [Chen et al. 2002].

4.1. Ambiente

Para os testes de performance, optamos por traces com o mesmo formato que os utilizados para os testes do Accural FD [Hayashibara et al. 2004], sendo que estes consideravam apenas duas máquinas. No nosso caso, a coleta dos traces foi realizada no PlanetLab [PlanetLab 2014] com dez sites, durante uma semana. Cada site enviou mensagens de *heartbeat* para os outros sites a uma taxa de um *heartbeat* a cada 100 ms (intervalo de envio). Sobre os *heartbeats* recebidos pelo site número *1* (considerado o nodo monitor), observamos que os tempos médios entre as chegadas dos *heartbeats* foi muito próximo a 100 ms. No entanto, em alguns sites, o desvio padrão é muito elevado, tal como no site *5* que tem desvio padrão de 310,958 ms com um mínimo de 0,006 ms e um máximo de 657.900,226 ms. Tais valores indicam que, durante um determinado intervalo de tempo da execução, o site parou de enviar *heartbeats* e começou novamente mais tarde. Notou-se também que o site *2* parou de enviar mensagens definitivamente depois de aproximadamente 48 horas. Os demais sites enviaram heartbeats durante todo o periodo de coleta. Finalmente, observamos que os sites *3* e *6* (respec. *5* e *8*) são os sites mais estáveis (respec. instáveis) com desvio padrão de 1,709 ms e 2,557 ms (respec. 310,96 ms e 100,714 ms).

A implementação do Impact FD para a avaliação foi baseada no Algoritmo 1 apresentado em [Rossetto et al. 2015]. Tal algoritmo emprega a abordagem baseada em tempo (*timer-based*) cujo valor do temporizador usa a estimativa de Chen (ver Seção 4.3). Além disso, considera-se um sistema assíncrono, com perda de mensagens em que a propriedade $PR(\diamondsuit IT)$ nao é satisfeita. Devemos ressaltar que, uma vez que os tempos de envio e chegada de cada *heartbeat* são registrados nos arquivos de traços, todos os experimentos foram realizados com exatamente os mesmos cenários e histórico de *heartbeats*.

4.2. Métricas de Qualidade (QoS)

Para avaliar o Impact FD, usamos três das métricas de QoS propostas por [Chen et al. 2002]: *tempo médio de detecção*, *taxa de erros média* e a *probabilidade de uma resposta precisa*. Considerando-se dois processos q e p onde p monitora q, a QoS do detector de falhas em p pode ser determinada a partir das transições entre os estados "trusted" e "not trusted" com relação a q.

- Tempo médio de detecção (T_D): é o tempo que decorre desde o momento em que o processo q falha até que o detector de falhas em p começa a suspeitar de q permanentemente;

Tabela 1. Configurações para o conjunto S

Configuração	Fator de impacto para cada site
Set 0	I_0=2; I_2=1; I_3=6; I_4=6; I_5=1; I_6=6; I_7=1; I_8=2; I_9=2;
Set 1	I_0=1; I_2=2; I_3=6; I_4=6; I_5=2; I_6=6; I_7=2; I_8=1; I_9=1;
Set 2	I_0=6; I_2=2; I_3=1; I_4=1; I_5=2; I_6=1; I_7=2; I_8=6; I_9=6;
Set 3	I_0=2; I_2=6; I_3=1; I_4=1; I_5=2; I_6=1; I_7=2; I_8=6; I_9=6;
Set 4	I_0=2; I_2=1; I_3=6; I_4=6; I_5=1; I_6=6; I_7=2; I_8=2; I_9=1;
Set 5	I_0=3; I_2=3; I_3=3; I_4=3; I_5=3; I_6=3; I_7=3; I_8=3; I_9=3;

- Taxa de erros média (λ_R): representa o número de erros que um detector de falhas faz em uma unidade de tempo, ou seja, a taxa com que o detector de falhas erra;
- Probabilidade de uma resposta precisa (P_A): é a probabilidade de que a saída do detector de falhas esteja correta num dado tempo aleatório.

4.3. Estimativa de chegada de *heartbeats*

Com o intuito de reduzir tanto o número de falsas suspeitas quanto o tempo para detectar uma falha, [Chen et al. 2002] propõem uma abordagem para estimar a chegada do próximo *heartbeat*, que também utilizamos nos nossos experimentos. Para fazer a estimativa, o protocolo se baseia no histórico dos tempos de chegada dos *heartbeats*. Assim, o temporizador é definido de acordo com esta estimativa acrescida de uma margem de segurança (β) constante. A estimativa é recalculada após a chegada de cada novo *heartbeat*. Já a margem de segurança é calculada uma única vez com base nos requisitos de QoS da aplicação.

Os autores sugerem que a margem de segurança β deve variar de 0 a 2500 ms. O tamanho da janela (*window size*) foi definido em 100 amostras para todos os experimentos, o que significa que o detector se baseia apenas nas últimas cem amostras de mensagens de *heartbeat* a fim de calcular a estimativa do tempo de chegada do próximo *heartbeat*.

4.4. Experimentos

Considerando os 10 sites dos traços, definimos que o conjunto S é composto por um único subconjunto com os sites 0, 2, 3, 4, 5, 6, 7, 8 e 9 (*S = {{0, 2, 3, 4, 5, 6, 7, 8, 9 } }*). O site *1* é o monitor (p) e não pertence a S. A Tabela 1 mostra as cinco configurações em relação aos valores do fator de impacto que consideramos para o conjunto S nos experimentos. Para todas as configurações, a soma de fator de impacto dos processos é 27.

As tres métricas propostas por [Chen et al. 2002] e descritas anteriormente foram avaliadas em três diferentes experimentos.

Experimento 1 - Probabilidade de uma resposta precisa: O objetivo deste experimento é avaliar a probabilidade de uma resposta precisa (P_A) com diferentes valores de *threshold* (18, 20, 21, 22, 23, 24 e 25) e diferentes configurações de fator de impacto (Tabela 1). Foi considerada a margem de segurança β = 400 ms.

A Figura 2 mostra que a P_A é menor quando o *threshold* aumenta. É importante lembrar que o *threshold* é um valor limite definido pelo usuário e se o *trust level* fornecido pelo Impact FD é igual ou maior que o *threshold*, a confiança no conjunto de processos

é garantida. Assim, os resultados confirmam que, quando o *threshold* é mais flexível, a probabilidade de uma resposta precisa é maior.

A configuração *"Set 0"* teve a maior P_A para todos os *threshold* devido à atribuição de um alto (respec. baixo) fator de impacto para os sites mais estáveis (respec. instáveis). Por outro lado, as configurações *"Set 2"* e *"Set 3"* tiveram a menor P_A uma vez que os sites instáveis estão com um fator de impacto alto nesses conjuntos. Por exemplo, para o site *8*, a média dos tempos entre as chegadas dos *heartbeats* recebidos é de 100,05 ms, mas o desvio padrão é de 100,71 ms. A atribuição do fator de impacto, portanto, tem influência sobre o desempenho da P_A.

Já a configuração *"Set 5"* apresenta uma queda abrupta da P_A quando o *threshold* é 25. Tal comportamento pode ser explicado uma vez que, nessa configuração de conjunto, todos os sites tem o mesmo fator de impacto (3). Portanto, cada falsa suspeita leva o *trust_level* ser menor que o *threshold* (25), aumentando a duração de erro e, consequentemente, diminuindo a P_A.

Observe, também, que o site *2* falhou após aproximadamente 48 horas. Assim, após a sua falha, a saída do Impact FD, que indica um *trust_level* menor que o *threshold*, não é um erro, ou seja, não é uma falsa suspeita. Por isso, em *"Set 3"*, cujo fator de impacto do site *2* é 6 (alto), a P_A é constante para *threshold* maior ou igual a 22: após a falha do site *2*, o *trust_level* fornecido pelo Impact FD é sempre menor que o *threshold* e falsas suspeitas relacionadas aos outros sites não o modificam. A duração média de erro do experimento é, portanto, menor após a falha, o que melhora a P_A.

Para a comparação da P_A do Impact FD com uma abordagem que monitora os processos individualmente, monitoramos cada site usando o mesmo algoritmo e parâmetros de [Chen et al. 2002] (*WS = 100*; β *= 400 ms*). Com isso, é possível avaliar a capacidade dos dois detectores de falhas (Impact e Chen) em reduzir o número de falsas respostas. A média da P_A obtida foi de 0,979788. Esse resultado mostra que, independentemente da configuração do conjunto, o Impact FD apresenta maior P_A do que [Chen et al. 2002] uma vez que o primeiro tem flexibilidade para tolerar falhas, ou seja, a duração de erro só começa a ser computada quando o *trust_level* fornecido pelo Impact FD é menor que o *threshold* definido, ao contrário do monitoramento individual, onde cada falsa suspeita aumenta a duração de erro.

Os resultados deste experimento destacam que a atribuição de diferentes valores de fator de impacto para os nodos pode degradar o desempenho do detector de falhas, especialmente quando os sites instáveis tem alto fator de impacto.

Experimento 2 - Tempo médio de detecção: No segundo experimento foi avaliada a probabilidade de uma resposta precisa (P_A) em relação ao tempo médio de detecção (T_D). Para este fim, variamos a margem de segurança (β) para obter diferentes valores de tempo de detecção. Esta então variou com intervalos de 100 ms, a partir de 100 ms. Neste experimento usamos a configuração *"Set 0"* e definimos diferentes *threshold*. Esta configuração de conjunto foi escolhida porque apresentou a melhor P_A para todos os *threshold* no Experimento 1. Também avaliamos a P_A e o T_D para o algoritmo de [Chen et al. 2002], o qual fornece como saída o conjunto de nodos suspeitos.

A Figura 3 mostra que para um *threshold* alto e tempo médio de detecção próximo a 200 ms, a P_A do Impact FD é menor, independentemente do *threshold*, isto porque a

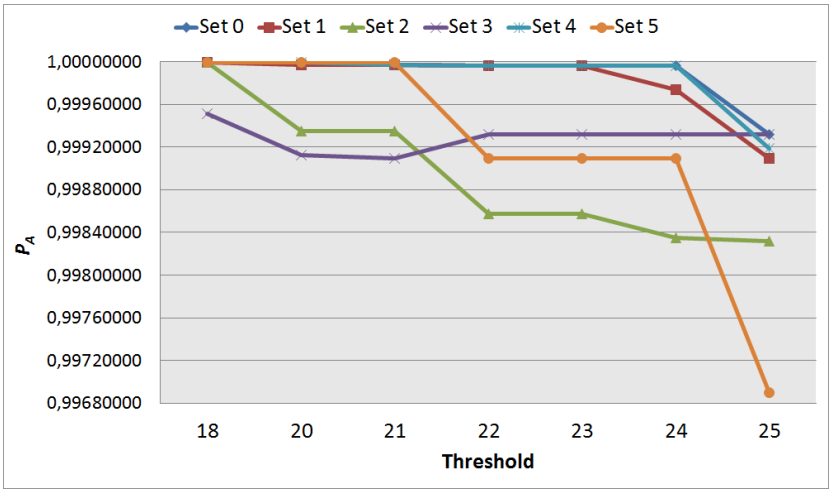


Figura 2. P_A **X threshold com diferentes configurações para o conjunto *S***

margem de segurança é, neste caso, igual a 100 ms, o que aumenta ambos o número de falsas suspeitas e a duração de erro. No entanto, quando o T_D é maior que 230 ms, a P_A do Impact FD é consideravelmente maior do que Chen. Após o tempo de detecção de aproximadamente 400 ms, a P_A do Impact FD torna-se constante, independentemente do tempo de detecção e *threshold*, ficando próxima a 1. Tal comportamento pode ser explicado, pois quanto maior a margem de segurança, menor é o número de falsas suspeitas e a duração de erro, o que confirma que quando o *timeout* é curto, as falhas são detectadas mais rapidamente, mas a probabilidade de ter falsas detecções aumenta [Satzger et al. 2007].

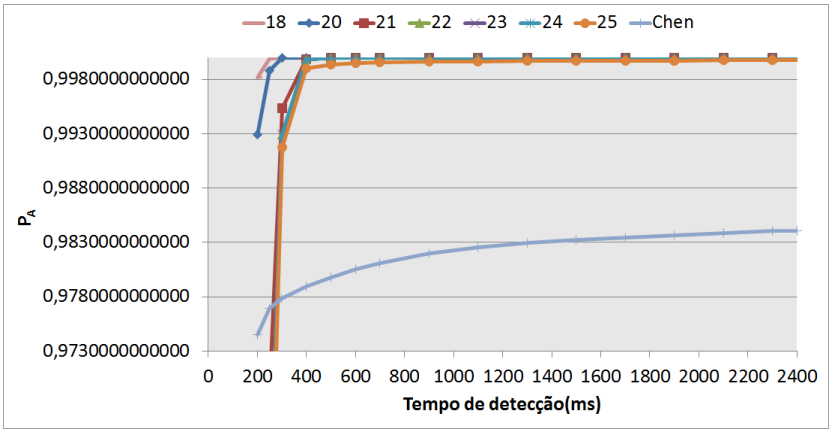


Figura 3. P_A **X** T_D **com diferentes *thresholds***

Experimento 3 - Taxa de erros média: Neste experimento avaliamos o tempo médio de detecção versus a taxa de erros média (erros por segundo). Consideramos a configuração *"Set 0"* e a taxa de erros é expressa em uma escala logarítmica. Podemos observar na Figura 4 que a taxa de erros média do Impact FD é maior quando o tempo de detecção é baixo (menor que 400 ms) e o *threshold* é alto (23-25). Tal resultado está de acordo com Experimento 2: sempre que a margem de segurança é pequena e o *threshold* tolera menos falhas, o Impact FD comete erros com mais freqüência. Em outras palavras, a taxa de erro diminui quando o *threshold* é mais flexível ou o tempo de detecção aumenta.

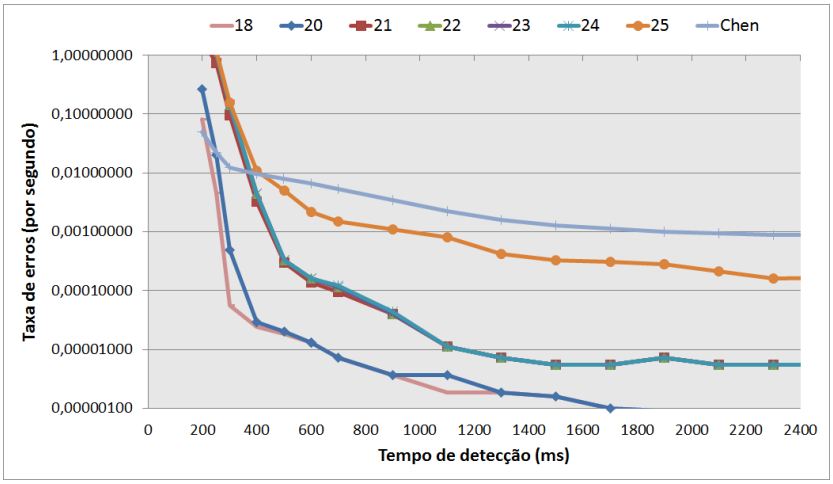


Figura 4. λ_R X T_D com diferentes *thresholds*

5. Trabalhos Relacionados

A maioria dos detectores de falhas não confiável da literatura é baseada no modelo binário e fornece como saída um conjunto de identificadores de processos, que, geralmente, expressa o conjunto de processos atualmente suspeitos de terem falhado ([Chandra and Toueg 1996], [Bertier et al. 2003]). Porém, em alguns detectores, como por exemplo os da classe Σ (respec. Ω) [Delporte-Gallet et al. 2004], o conjunto de processos (respec., o processo) de saída refere-se aos processos não suspeitos de estarem falhos ($trusted$).

O detector de falhas Accrual ϕ [Hayashibara et al. 2004] propõe uma abordagem em que a saída do detector é, para cada processo, o nível de suspeita em uma escala contínua, em vez de fornecer informações de natureza binária (*trusted* ou *suspected*). Assim, ele tem por objetivo dissociar o monitoramento da interpretação. O nível de suspeita do processo q, monitorado por p, expressa o grau de confiança de p na asserção de que q falhou. Se este realmente falhou, o valor do nível de suspeita acumula ao longo do tempo e, consequentemente, tende ao infinito. Em [Satzger et al. 2007], os autores apresentam uma nova abordagem de Accrual FD com base em uma estimativa acerca da densidade do histograma de tempo de chegada amostrado. Levando em consideração uma amostra dos tempos entre chegadas, bem como o tempo do último *heartbeat* recebido, o algoritmo calcula a probabilidade de que não haja mais mensagens de *heartbeat* para chegar, ou seja, o processo falhou.

[Leners et al. 2013] é um trabalho recente que propõe um serviço para reportar falhas às aplicações, encapsulando a incerteza. Ele parte da premissa de que os aplicativos deveriam ter informações sobre as falhas para tomar ações de recuperação adequadas. O objetivo é fornecer relatórios de estado vinculados à detecção de falhas com uma abstração que relata o grau de certeza. O trabalho fornecido em [Brun et al. 2011] apresenta uma nova técnica de redundância, baseada em votação, a fim de melhorar a confiabilidade em sistemas distribuídos, considerando que cada nodo tem uma probabilidade de ser byzantino. Considerando tais valores, os autores calculam o número mínimo de máquinas necessário para que o sistema como um todo ofereça confiabilidade igual ou maior a um limiar (*threshold*).

6. Conclusão e Trabalhos Futuros

Neste artigo, apresentamos e definimos o detector de falhas não confiável Impact FD, que fornece uma saída relacionada com um conjunto de processos e não apenas para cada um individualmente. Ambos o *fator de impacto* e o *threshold* oferecem um grau de flexibilidade já que permitem ao usuário ajustar o Impact FD de acordo com as necessidades específicas e a margem de falhas aceitável da aplicação. Em alguns cenários e configurações, eles também podem reduzir a taxa de falsas respostas quando comparado com detectores de falhas não confiáveis tradicionais. Os resultados da avaliação de desempenho mostram que a atribuição de um alto (respec. baixo) fator de impacto para os nodos mais estáveis (respec. instáveis) aumenta a probabilidade de uma resposta precisa do detector de falhas.

Como trabalho futuro, pretende-se estender o Impact FD a fim de abordar as falhas de comportamento impróprio (*misbehavior*). Também estamos trabalhando na redução e equivalência do Impact FD em relação a outros detectores (por exemplo, Sigma e Ômega [Delporte-Gallet et al. 2004]), o que torna necessário algumas condições e/ou domínios específicos como hipótese. Outra direcão de nossa pesquisa é a realização de outros experimentos em redes diferentes, tais como Wi-Fi ou rede local.

Referências

Bertier, M., Marin, O., Sens, P., et al. (2003). Performance analysis of a hierarchical failure detector. In *DSN*, volume 3, pages 635–644.

Brun, Y., Edwards, G., Bang, J. Y., and Medvidovic, N. (2011). Smart redundancy for distributed computation. In *ICDCS)*, pages 665–676.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267.

Chen, W., Toueg, S., and Aguilera, M. K. (2002). On the quality of service of failure detectors. *Computers, IEEE Transactions on*, 51(5):561–580.

Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Hadzilacos, V., Kouznetsov, P., and Toueg, S. (2004). The weakest failure detectors to solve certain fundamental problems in distributed computing. In *ACM PODC*, pages 338–346.

Hayashibara, N., Defago, X., Yared, R., and Katayama, T. (2004). The φ accrual failure detector. In *SRDS*, pages 66–78.

Leners, J. B., Gupta, T., Aguilera, M. K., and Walfish, M. (2013). Improving availability in distributed systems with failure informers. In *NSDI*.

Mostefaoui, A., Mourgaya, E., and Raynal, M. (2003). Asynchronous implementation of failure detectors. In *DSN*, pages 351–351.

PlanetLab (2014). Planetlab. `http://www.planet-lab.org`.

Rossetto, A. G., Geyer, C. F., Arantes, L., and Sens, P. (2015). Impact: an unreliable failure detector based on processes' relevance and the confidence degree in the system. Technical report. INRIA Nº hal-01136595.

Satzger, B., Pietzowski, A., Trumler, W., and Ungerer, T. (2007). A new adaptive accrual failure detector for dependable distributed systems. In *ACM SAC*, pages 551–555.

CRAL: um algoritmo de roteamento baseado em centralidade e energia para Redes de Sensores Sem Fio

Bruno P. Santos[1], **Luiz F. M. Vieira**[1], **Marcos A. M. Vieira**[1]

[1] Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
31270-010 – Belo Horizonte – MG – Brasil

{bruno.ps, lfvieira, mmvieira}@dcc.ufmg.br

Abstract. *High throughput and energy are two important constraints of the Wireless Sensor Networks (WSNs). In this paper, we present the Centrality-based Routing Aware for L2Ns (CRAL)*[1]*-Low Drop and Fast Delivery to mitigate these problems. The routing protocols are centrality-based and employ link quality estimators Expected Transmission Count (ETX) and Expected Transmission Time (ETT) to find best paths in wireless links. The suitable combinations of these techniques lead the algorithms to improve the literature results in delivery rate, energy consumption, and time to delivery data packets. CRAL does this by building routing trees with high throughput and maintains low energy consumption. The simulation results show that CRAL is more reliable, efficient in energy consumption, robust, and favoring data fusion than Centrality Tree (CT) and Shortest Paths Tree (SPT).*

Resumo. *Vazão dos dados e energia disponível são algumas das principais limitações das Redes de Sensores Sem Fio (RSSF). Este artigo descreve os algoritmos Centrality-based Routing Aware for L2Ns (CRAL)-Low Drop e Fast Delivery para mitigar esses problemas. Os algoritmos são baseados em centralidade Sink Betweenness e métricas Expected Transmission Count (ETX) e Expected Transmission Time (ETT). A combinação apropriada dessas técnicas leva os algoritmos a melhorarem as soluções existentes em termos de taxa de entrega, consumo de energia e tempo de entrega das mensagens de dados. CRAL faz isso através a árvores de roteamento de alta vazão, ao passo que mantém um baixo consumo de energia. Os resultados de simulação mostram que os algoritmos CRALs são mais robustos, eficientes no consumo de energia e favorecem a fusão de dados do que Centrality Tree (CT) e Shortest Paths Tree (SPT).*

1. Introdução

Eventos em sistemas computacionais são entidades de programação que representam um acontecimento do mundo real ou não [Boukerche et al. 2014]. Como exemplo de sistema de processamento de eventos destacam-se as Redes de Sensores Sem Fios (RSSFs), as quais têm por finalidade coletar, analisar e reagir adequadamente aos eventos. As fases de coleta e análise dos eventos apresentam vários desafios como criar rotas, reduzir o tempo de transmissão e agregar dados, entre outros [Ramos et al. 2014, Boukerche et al. 2014]. Na fase de coleta, questões pertinentes são: como criar uma estrutura de roteamento?

[1] Low power and Lossy Networks.

Quais são os modelos mais apropriados para as características da rede sem fio? Já na fase de análise, como aplicar técnicas de fusão de dados, mineração de dados, aprendizado de máquinas, dentre outras para que as RSSF sejam eficientes e poupem recursos?

RSSF geralmente são dirigidas a eventos e toda a rede coopera para que eventos sensoreado pelos nós alcancem seu destino. Esta tarefa é desafiadora em face das severas limitações computacionais, de comunicação, e, em especial, de energia dos dispositivos empregados nessas redes. Portanto, modelos regulados especialmente para estes sistemas devem ser propostos. O roteamento influi diretamente no consumo dos recursos dos dispositivos e é fundamental para realizar análise de dados em tempo de roteamento. Por isso, este trabalho trata do problema de roteamento de mensagens em redes de sensores sem fio. Neste sentido, são apresentadas técnicas para ajustar métricas adequadas para o ambiente das RSSF combinado com a importância topológica dos nós na rede para construção da estrutura de coleta dos dados. Os resultados de simulação mostram que a combinação apropriada dessas técnicas conduz a árvores de roteamento eficientes em dois quesitos: alta vazão e consumo de energia. Além disso, as estratégias apresentadas se mostraram mais favoráveis à fusão de dados e apresentam baixo número de transmissões para entregar os dados, quando comparadas com outras soluções.

Em resumo, as contribuições deste artigo incluem: 1) proposta do *Centrality-based Routing Aware for L2Ns (CRAL)-Low Drop* e *Fast Delivery* como protocolos de roteamento baseado em centralidade e qualidade do enlace para RSSF; 2) proposta do uso de diferentes funções de fusão de dados em conjunto com CRAL-*Low Drop* (CRAL-LD) e CRAL-*Fast Delivery* (CRAL-FD).

Os algoritmos aqui propostos se diferem dos apresentados na literatura por um ou dois fatores. *Collection Tree Protocol (CTP)*, por exemplo, não faz uso da importância topológica dos nós para criar rotas, enquanto *Centrality Tree (CT)* não emprega *Link Quality Estimation (LQE)* para escolher as melhores rotas sem fio. Deste modo, *CRAL* leva em conta esses fatores para prover rotas de alta qualidade ao passo que cria rotas com pontos intermediários comuns, o que facilita a análise de dados. CRAL-LD e CRAL-FD se diferenciam no método utilizado para qualificar as rotas, em que empregam as métricas *Expected Transmission Count (ETX)* e *Expected Transmission Time (ETT)* respectivamente.

Na próxima seção são discutidos os principais trabalhos relacionados. Na Seção 3, são discutidos os detalhes do problema abordado. Logo após, na Seção 4, são exibidos os detalhes da arquitetura e algoritmo CRAL. Na Seção 5, uma avaliação é realizada para mostrar o desempenho dos protocolos CRAL-LD e CRAL-FD comparados entre si e em relação ao estado da arte. Na Seção 6, as considerações finais são apresentadas.

2. Trabalhos Relacionados

Tabela 1. Comparação dos algoritmos de roteamento

	SPBC	TLC	FBC	RBC	CT	**CRAL-LD**	**CRAL-FD**	CTP	SPT	CNS	InFRA	DAARP
Centralidade	BC	BC	BC	BC	SBC	SBC	SBC	–	–	–	–	–
Cod. distribuído	–	–	–	–	✓	✓	✓	✓	✓	✓	✓	✓
LQE	–	–	–	–	–	✓	✓	✓	–	–	–	–

BC – Betweenness Centrality
SBC – Sink Betweenness Centrality

Na Tabela 1 são classificados os algoritmos de roteamento de acordo com as características a seguir: i) aplicação de centralidade na construção da árvore de roteamento;

ii) uso de LQE para escolher enlaces de melhor qualidade. iii) código implementado de forma distribuída ou centralizada. Neste sentido, este artigo apresenta as bases de como construir os algoritmos CRAL-*Low Drop* e *Fast Delivery*, os quais são distribuídos e usam combinações de técnicas de centralidade e LQEs.

Shortest Path Betweenness-Centrality (SPBC) e *Traffic Load Centrality (TLC)* são os algoritmos de roteamento baseado na centralidade *Betweenness* mais tradicionais. Nestas abordagens, somente os menores caminhos são usados para transportar os dados. TLC difere de SPBC pois utiliza o tráfego para ponderar as rotas. Um dos problemas dessas estratégias de roteamento reside na utilização os menores caminhos, tornando-as estratégias inflexíveis para balanceamento de carga, tolerância a falhas ou acordo de níveis de serviço [Dolev et al. 2010]. *Flow Betweenness-Centrality (FBC)*, ao contrário de SPBC e TLC, assume que as rotas são simples (não possuem ciclos) e considera igualmente caminhos de todos os comprimentos. FBC é razoável ao assumir rotas simples, porém FBC falha ao considerar todas as rotas igualmente, pois em redes é preferível caminhos mais curtos em detrimento dos longos. Em [Dolev et al. 2010] é apresentado o *Routing Betweenness Centrality (RBC)* como uma generalização das estratégias SPBC, TLC, e FBC. Contudo, os autores de dos algoritmos não foram projetados de modo distribuído como CRAL-LD e CRAL-FD, portanto o cálculo da centralidade deve ser realizado antes da execução do roteamento. Além disso, não são consideradas as restrições impostas pelas RSSF como, por exemplo, taxa de perda nos enlaces da rede e não é realizada uma avaliação do consumo de energia dos nós.

Dos algoritmos não são baseados em centralidade e que não usam LQE apresentados na Tabela 1, tem-se *Shortest Paths Tree (SPT)* e *Center at Nearest Source (CNS)* como representantes tradicionais. Para SPT, os menores caminhos são utilizados para encaminhar todo o fluxo da rede. CNS escolhe um nó mais próximo do sorvedouro recebe mensagens dos descendentes e encaminha os dados para o sorvedouro. *Information Fusion-based Role Assignment (InFRA)* e *Data-Aggregation Aware Routing Protocol (DAARP)* são discutidos em [Villas et al. 2010]. InFRA e DAARP são aproximações da árvore de Steiner [Fasolo et al. 2007] e foram otimizados para eventos que ocorrem de forma aglomerada (*clustering*), isto é, os eventos ocorrem em poucos pontos da região monitorada e somente os nós com uma distância S do evento reportam os dados. O CRAL é otimizado para sensoreamento periódico que generaliza a anterior. Desta forma, é permitido que eventos ocorram em locais aleatórios da região monitorada. Além disso, CRAL usa centralidade para escolher nós intermediários na árvore de roteamento, enquanto InFRA e DAARP fazem o roteamento baseado em cada grupo (*cluster*) de evento detectado.

A qualidade do enlace é um quesito que gera grande impacto no desempenho da RSSF [Baccour et al. 2009]. O consumo de energia, tempo e taxa de entrega dos dados sofrem quando a rota é de baixa qualidade. CTP usa um LQE denominado *ETX*, específico para redes sem fio. Algoritmos como RBC e CT [Oliveira et al. 2010] usam centralidade para construir suas rotas, porém não consideram a qualidade dos enlaces. CRAL-LD, ao contrário dos anteriores, além de utilizar centralidade *Betweenness* para escolher os melhores nós repetidores no meio da rede, emprega ETX para encontrar caminhos de qualidade, minimizar o consumo de energia e efeito *overhearing*. CRAL-FD utiliza *ETT*, que considera não só o ETX como também a taxa de transmissão do rádio, assim rotas são configuradas para obter alta vazão e baixa perda de pacotes.

3. Problema do roteamento em RSSF

Roteamento em RSSF enfrenta compromissos impostos por restrições energéticas, computacionais e de comunicação. Geralmente quatro princípios são tidos como fundamentais e os algoritmo de roteamento devem buscar o equilíbrio entre eles: 1) **confiabilidade**, o protocolo deve entregar a maior quantidade dos pacotes possível, quando existe rota entre as entidades da comunicação; 2) **robustez**, o protocolo deve operar em diferentes topologias, cargas, quantidade de nós e na presença de falhas. 3) **eficiência**, em que deve-se entregar os pacotes com a menor número de transmissões e tempo possível, economizar energia e manter a menor quantidade de estados possível.

Coleta dados de em RSSF de modo confiável, robusto e eficiente pode ser realizada com o protocolo CRAL, o qual balanceia os compromissos impostos pelos princípios das RSSF. Para realizar esta tarefa, CRAL combina inteligentemente centralidade com modelos regulados para ambientes sem fio, os quais serão discutidos nas próximas seções.

3.1. Centralidade

A árvore de Steiner é conhecida em redes de computadores por suas diferentes aplicações. Para RSSF, esta árvore apresenta pontos ótimos para a fusão de dados, o que minimiza o consumo energético [Fasolo et al. 2007]. Entretanto, o problema de encontrar a árvore de Steiner é NP-Difícil, assim soluções aproximadas devem ser propostas.

A posição topológica do nó influencia seu potencial controle sobre os dados que fluem na rede [Dolev et al. 2010]. A centralidade *Betweenness* é uma medida que quantifica a importância topológica de um nó [Almiron et al. 2010]. Um alto valor de *Betweenness* de um nó, em uma RSSF, indica que o nó roteia grande parte dos pacotes da rede. A informação da centralidade dos nós é um dos objetos chave deste trabalho, pois baseado nesta informação é possível construir uma árvore de roteamento especial, em que os nós intermediários (mais centrais) são pontos comuns às rotas da maioria dos nós. Usando os nós centrais a quantidade de intermediários é reduzida e pontos de fusão de dados são favorecidos. Isto coloca eleva esta estratégia a uma aproximação da árvore de Steiner.

Sink Betweenness Centrality (SBC) é uma métrica baseada no *Betweenness* para coleta de dados em RSSF [Almiron et al. 2010, Oliveira et al. 2010]. O computar o SBC de um nó t depende do número de caminhos partindo dele até o sorvedouro e, do número de suas participações, como intermediário, nos menores caminhos dos seus descendentes até o sorvedouro. Formalmente, seja $T = (V, E)$ uma árvore, onde V é um conjunto de vértices e E é um conjunto de arestas, além disso, seja $s \in V$ o sorvedouro, então o SBC de um nó t é definido como $SBC_t = \sum_{i \in \psi_t} \frac{\sigma_{ts}}{\sigma_{is}}$, onde σ_{ts} é o número de menores caminhos de t até s, σ_{is} é o número de menores caminhos de i até s, uma vez que $\psi_t = \{i \in V | t \in SP_{i \to s}\}$ é o conjunto de todos os menores caminhos de i até s ($SP_{i \to s}$) que contém t como um nó intermediário em, ao menos, um de seus menores caminhos.

A Figura 1 mostra o SBC para cada vértice de um grafo. Para calcular SBC_A é necessário saber que a quantidade de rotas partindo de A até S é 1 (numerador da equação) e o número de participações de A nas menores rotas dos vértices B, C, D que são respectivamente $1, 1, 2$ (denominadores da equação). Portanto, $SBC_A = \frac{1}{1} + \frac{1}{1} + \frac{1}{2}$. O raciocínio é o mesmo para encontrar o SBC dos demais vértices. Note que S possui 0 rotas e D não faz parte do menor caminho de nenhum outro vértice, portanto o $SBC_{S,D} = 0$.

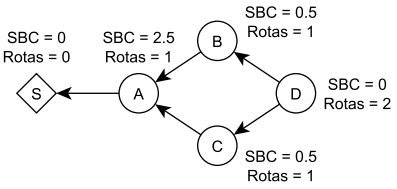


Figura 1. Sink Betweenness Centrality para cada vértice do grafo.

3.2. Qualidade do enlace sem fio

LQEs está sendo objeto de estudo dos últimos anos [Baccour et al. 2009, Baccour et al. 2012, Javaid et al. 2009]. Todo este esforço é explicado pelo grande impacto desta informação no desempenho da rede sem fio, além de afetar o projeto dos protocolos das camadas superiores. Em [De Couto et al. 2005] é apresentada a métrica ETX, a qual é uma aproximação mais apropriada para escolher enlaces sem fio. ETX minimiza o número esperado de transmissões e retransmissões para entregar os pacotes. *Expected Transmission Time (ETT)* [Draves et al. 2008] é uma variação mantém todas as características do ETX e vai além possibilitando escolher enlaces baseado-se na taxa de transmissão dados, isto é, ETT considera enlaces com tempos de transmissão diferentes.

4. Centrality-based Routing Aware for L2Ns

CRAL é um algoritmo de roteamento que seleciona nós intermediários usando o SBC e considera que os enlaces possuem taxa intermediária de perda, além de diferentes taxas de transmissão. Neste sentido, são apresentadas duas versões do protocolo: 1) CRAL-*Low Drop* que emprega ETX como LQE para minimizar a quantidade de perda de pacotes; 2) CRAL-*Fast Delivery* que usa ETT para estimar os enlaces, desta forma são escolhidos enlaces para reduzir o tempo de entrega e perda de pacotes. É assumido que ETT irá operar sobre rádios com multiplas taxas de transmissão, por exemplo, rádios operando com o padrão IEEE 802.11. Similar à maioria dos protocolos de coleta de dados, CRAL começa pela requisição do sorvedouro. Esta requisição agendará o estabelecimento da árvore de rotas usando estimadores ETX e ETT. Ao receber a requisição os nós enviam uma resposta de volta para o sorvedouro. Deste modo, cada nó pode deduzir quantos filhos possui e seus caminhos, o que permite calcular do SBC.

4.1. Arquitetura

A arquitetura do CRAL-LD e CRAL-FD é semelhante exceto pelo estimador de enlaces utilizado. A Figura 2 exibe os principais módulos e seus relacionamentos. A arquitetura é dividida em plano de dados e plano de controle. No plano de controle, tem-se o Módulo de Rotas e Centralidade (MRC) responsável pelo preenchimento da Tabela de Rotas, as rotas são escolhidas conforme as informações do Estimador de Enlace (EE) e da *Sink Betweenness Centrality* dos nós. O EE obtém informações dos enlaces por consultar a camada de enlace de dados. No plano de dados, o Módulo de Encaminhamento (ME) que é responsável pela recepção e encaminhamento das mensagens. Caso seja interceptada uma mensagem de controle, o ME sinaliza ao MRC para que as decisões apropriadas sejam tomadas. Em caso de mensagens de dados, o ME consulta a Tabela de Rotas para encaminhar o pacote ao próximo salto da rota até o sorvedouro. Na Figura 2 também é mostrado

um módulo de interface que permite a comunicação inter-camadas. E através desta interface, foram implementadas as funções de fusão de dados descritas na Seção 4.3).

CRAL usa quatro tipos de mensagens de controle: 1) Mensagem de Descoberta (MD), o qual que contém os campos ID do nó, P número de caminhos, H número de saltos, e RQ qualidade da rota (ETX da rota); 2) Mensagem de Resposta (MR) apresenta os mesmos campos da MD, porém adiciona o campo *SBC* para representar a centralidade; 3) Mensagem de Requisição de Rota (MRR) e Mensagem de Requisição de Desligamento (MRD) necessitam somente do campo ID do nó;

4.2. Algoritmo

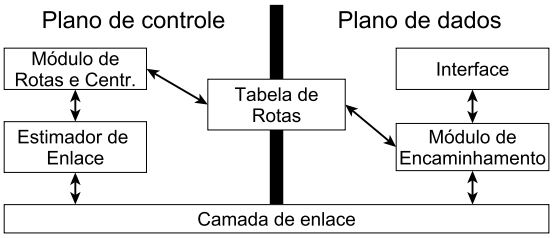


Figura 2. Arquitetura dos algoritmos CRALs.

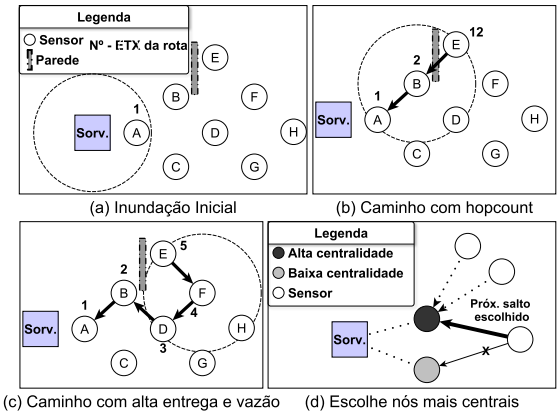


Figura 3. Operação dos algoritmos CRALs.

Descoberta e Resposta são as duas principais fases do algoritmo CRAL. Na primeira fase, o intuito é encontrar os caminhos com melhores enlaces segundo os estimadores ETX e ETT, além de encontrar o número de caminhos entre um nó e o sorvedouro. O Algoritmo 1 e a Figura 3 (a)(b)(c) ilustram essa fase, que tem seu início quando o sorvedouro inunda a rede com a MD (linha 8 Algoritmo 1 e Figura 3(a)). Os nós ao receberem a mensagem de descoberta, anotam as informações do melhor enlace, número de saltos e o número de rotas de melhor qualidade até o sorvedouro (linhas 10 a 15 e Figura 3(b)). No final desta fase obtemos o numerador da equação do SBC (Seção 3.1). E ao usar RQ baseado no ETX ou ETT o caminho da Figura 3(c) é favorecido.

Na fase de Resposta, através dos caminhos encontrados, os nós enviam informações computadas na fase inicial e seu valor de centralidade. O cálculo do SBC e escolha dos nós mais centrais são os objetivos dessa fase (Algoritmo 2). Esta fase inicia na linha 17 do Algoritmo 1. No momento que uma MR é interceptada (linha 1 Algoritmo 2) por um nó intermediário t, este possui as informações necessárias para o cálculo do SBC, isto é, o número de rotas de t e o número de participações de t nas rotas dos seus descendentes. Após o cálculo do SBC (linhas 4) o nó intermediário atualiza a mensagem MR com suas informações de centralidade, número de rotas e qualidade da rota e encaminha a MR para o sorvedouro (linha 6). Um nó t ao receber uma MR verifica se a mensagem recebida veio de um nó x mais próximo do sorvedouro do que t (linha 8) e se x é o mais central (linha 9). Em caso afirmativo, o nó t escolhe x como próximo salto (linha 9 do Algoritmo 2 e Figura 3(d)).

Algoritmo 1: Fase Descoberta

```
[1]  Sorv ← 1; Caminhos_t ← 1;
[2]  SBC_t ← 0; MAX_SBC ← ∅;
[3]  Saltos_t ← ∞; RQ_t ← ∞;
[4]  PróxSalto_t ← ∅;
[5]  se Sorv então
[6]      MD_H ← Saltos;
[7]      MD_P ← Caminhos;
[8]      Broadcast MD
[9]  senão se Recebeu MD então
[10]     se MD_RQ < RQ_t então
[11]         //Melhor rota
[12]     fim
[13]     se MD_RQ = RQ_t então
[14]         /*Rotas de mesma
             qualidade*/
[15]     fim
[16]     Escalona Broadcast MD;
[17]     Escalona Broadcast MR;
[18] fim
```

Algoritmo 2: Fase Resposta

```
[1]  se Recebeu MR então
[2]      se MR_RQ > RQ_t então
[3]          //MR de um nó filho
[4]          Efetua cálculo do SBC;
[5]          //Atualiza MR
[6]          Broadcast MR;
[7]      fim
[8]      se
         MR_RQ + RQ_{t→MR_ID} ≤ RQ_t
         então
[9]          se MR_SBC > Max_SBC
             então
[10]             MAX_SBC ← MR_SBC;
[11]             PróxSalto_t ← MR_ID;
[12]         fim
[13]     fim
[14] fim
```

A entrada e saída dos nós na rede são rotinas simples. Na primeira, o nó t envia para seus vizinhos diretos uma MRR. Os nós ao receberem MRRs, enviam uma MD com suas informações atualizadas, isto irá disparar o ciclo de atualização de rotas abordadas nos Algoritmos 1 e 2, como consequência o recálculo do SBC e escolha do nó mais central. Ao se desligar da rede, um nó envia uma MRD. Os nós filhos diretos ao receberem MRD dispara uma mensagem MRR para o estabelecimento de sua nova rota.

4.3. Funções de fusão de dados

A principal ideia da fusão de dados em RSSF é enviar mensagens agregadas, ao invés de enviar individualmente as mensagens dos nós para o sorvedouro. Isto faz da fusão de dados um mecanismo poderoso para reduzir o número de transmissões e consequentemente o consumo de energia da rede. Este trabalho foca em fusão de dados para RSSF no cenário em que os dados que são produzidos periodicamente. Neste sentido, três tipos de fusão de dados são as mais empregadas na literatura [Solis and Obraczka 2003], sendo elas: 1) *Simple*, em que um nó agrega as mensagens que chegam durante um tempo t. *Simple* tem fácil implementação, porém mensagens oriundas dos nós folha levam $t \times saltos$ para serem entregues ao sorvedouro. 2) *Periodic per hop simple (PerHop)*, os nós também esperam uma pré-definida quantidade de tempo para fundir as mensagens, porém o tempo depende da posição do nó na árvore de rotas. A vantagem desta técnica sobre *simple* é a redução do tempo de pior caso. 3) *Periodic per hop adjusted (PerHopAd)*, os nós esperam um tempo t que depende da sua distância (em tempo) até o sorvedouro. Além disso, a função dispara a mensagem agregada assim que uma mensagem de cada nó descendente é recebida. O tempo de espera faz com que ocorra um efeito de cascata, em que os nós próximos ao sorvedouro levam mais tempo agregando mensagens ao passo

que nós próximos as folhas da árvore levam menos tempo. Assim é possível reduzir ainda mais o tempo de entrega das mensagens agregadas.

4.4. Discussão

Manutenção da escolha dos enlaces é uma questão pertinente do algoritmo CRAL. Os estimadores de enlace ETX e ETT são baseados em software [Baccour et al. 2012] e necessitam que mensagens de sonda sejam envidadas, por isso o protocolo CRAL assume a existência do módulo EE para cálculo do LQE. O EE enfrenta um compromisso entre estabelecimento/manutenção das rotas e o consumo energético por enviar sonda. Para equilibrar o compromisso, CTP emprega o algoritmo *Trickle* [Levis et al. 2003], o qual envia poucas sondas quando a rede é estável e aumenta o número de sondas quando a rede é instável reagindo às mudanças topológicas. CRAL também pode operar em conjunto com o *Trickle*, visto que a arquitetura permite anexar diferentes algoritmos para controlar o EE (vide Figura 2).

Como mitigar o problema dos nós mais centrais e próximos ao sorvedouro consumirem mais energia? Uma maneira seria efetuar o balanceamento do tráfego entre os nós com maior centralidade e os com baixa centralidade. Alternativamente rotear baseando-se na energia residual dos nós é possível. Uma terceira abordagem é o uso de técnicas *energy hole* [Li and Mohapatra 2005].

5. Avaliação

Nesta seção serão analisados os algoritmos CRAL-LD e CRAL-FD em comparação com duas outras abordagens da literatura: CT e SPT[2] (como referência). Dos algoritmos apresentados na Tabela 1, RBC e seus derivados não foram projetados para RSSF, portanto não podem ser comparados com CRAL-LD e CRAL-FD. InFRA e DAARP são de classes distintas de algoritmos de coleta de dados, eles não apresentam resultados satisfatórios quando eventos ocorrem de forma não aglomerada. CTP não utiliza centralidade e não foi projetado para operar com fusão de dados. CNS é um algoritmo clássico e apresentas resultados inferiores aos apresentados pelos algoritmos CT e SPT.

As análises aqui realizadas têm como objetivos principais validar os algoritmos propostos e avaliar o desempenho dos algoritmos CRAL-LD e CRAL-FD em relação ao estado da arte. Foram utilizadas as seguintes métricas para comparação: 1) Consumo de energia; 2) Número de Steiner nodes; 3) Eficiência (número transmissões para entregar as mensagens e taxa de entrega); 4) *Overhearing*; 5) Latência; 6) Qualidade da rota; 7) *Freshness* (velocidade de entrega das mensagens agregadas).

5.1. Cenários de Simulação e Métricas Utilizadas

O cenário padrão usado nos resultados é exibido na Tabela 4, para alguns resultados, os parâmetros podem ser variados e isso é descrito no conforme necessário. Para avaliar a escalabilidade, foram usadas redes de diferentes portes: pequeno (128 nós), médio (256 e 512 nós), e grande (1024 nós) dispostos na região de interesse aleatóriamente. Foram executados experimentos em 33 topologias diferentes (em cada porte da rede), em cada uma das topologias realizou-se 33 simulações, totalizando 1089 simulações por

[2]SPT não é exibido em alguns resultados, pois apresenta valores equivalentes ou inferiores aos do CT, o qual calcula o SPT internamente.

Figura 4. Parâmetros de Simulação

Parâmetro	Valor
Sorvedouro	1 borda
Nº de nós sensores	1024
Rádio Padrão	$802.11b$
Taxa de perda no enlace (valor/aresta)	$0 \leq p \leq 1$
Densidade (*nós*/m^2)	20
Mensagens de dados (mensagens/nós)	20
Duração do envio das mensagens de dados (s)	20
Nº padrão de retransmissões	10

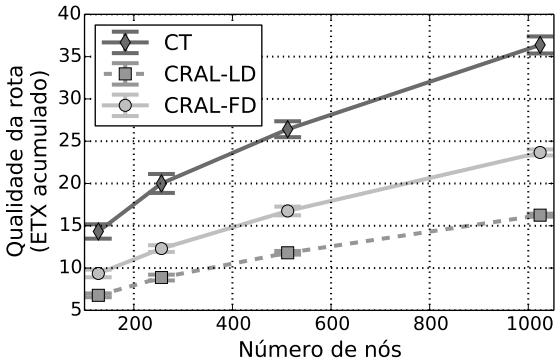


Figura 5. ETX médio das rotas variando número de nós sensores.

porte. Vale ressaltar que os parâmetros referentes ao tráfego de dados foram customizados para o envio periódico de mensagens. O modelo multi-taxa utilizado neste trabalho é baseado no padrão [IEEE 802.11b 1999]. Os parâmetros de alcance são baseados nas especificações do rádio Lucent ORiNOCO PC Card [Orinoco wireless networks 2014], assim é possível avaliar também a métrica ETT que considera diferentes taxas de transmissão. Em cada gráfico, a curva representa os valores médios e as barras de erro representa o intervalo de confiança de 95%. Todos os algoritmos foram implementados no simulador Sinalgo $v.0.75.3$ [Sinalgo 2014].

5.2. Resultados de simulação

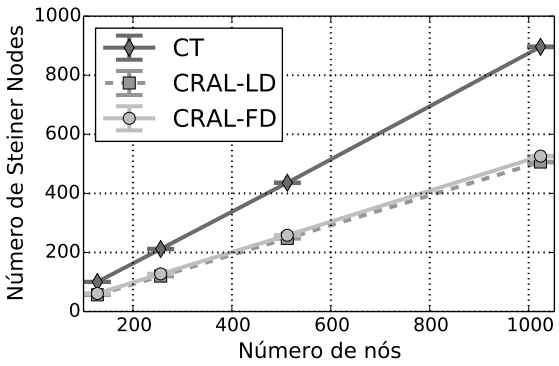


Figura 6. Steiner Nodes variando o número de nós sensores.

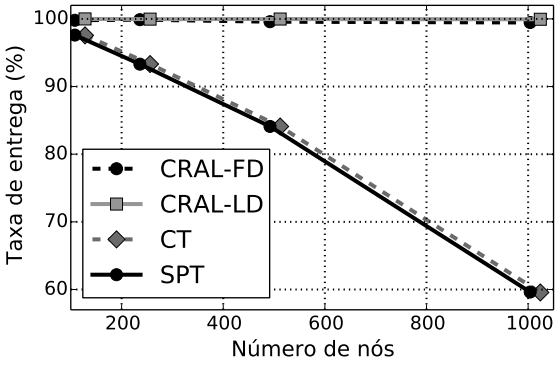
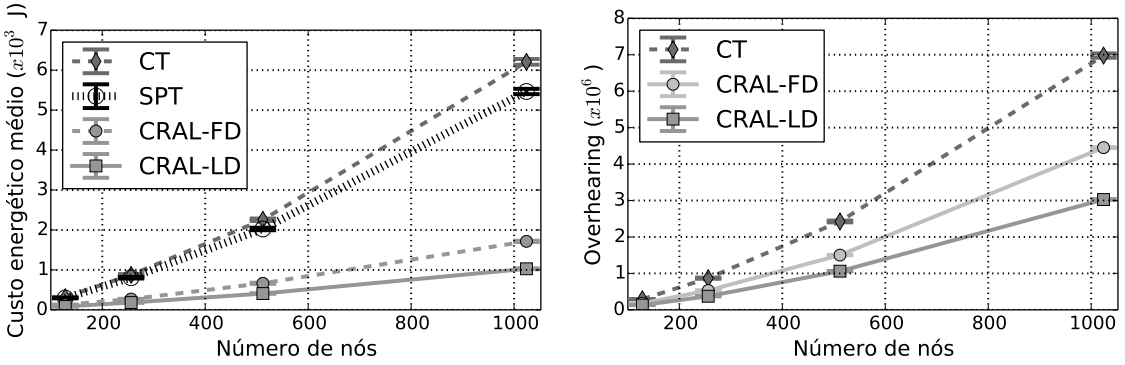


Figura 7. Taxa de entrega das mensagens de dados.

Inicialmente o ETX médio foi calculado para mostrar a qualidade das rotas, este valor indica o número médio requerido de transmissões (incluindo retransmissões) para rotear os dados até o sorvedouro. A Figura 5 mostra o ETX médio das rotas encontradas por CRAL-LD, CRAL-FD, CT e SPT. O protocolo CRAL-LD encontra rotas com menor número de transmissões para entregar os dados, CRAL-FD valores maiores que CRAL-LD, pois prioriza rotas com alta vazão além da qualidade, isto geralmente conduz a maior quantidade de saltos. CT em todos os casos apresenta rotas de baixa qualidade por usar o *hop count* como métrica de rota. Um baixo número de nós Steiner implica em uma árvore para fusão de dados melhores e que economia de energia. A Figura 6 mostra o número de nós Steiner para cada protocolo. CT apresenta maior quantidade de nós

Steiner do que CRAL-LD e CRAL-FD em todos os cenários considerados. A diferença é intensificada em redes de grande porte, em que CRAL-LD e CRAL-FD apresentam aproximadamente metade dos nós Steiner do que CT.

A taxa de entrega de dados foi verificada variando o porte da rede. Neste experimento foi habilitada uma quantidade máxima de retransmissões do pacote por saltos (Tabela 4). A Figura 7 exibe os resultados para este experimento. As abordagens CRAL-LD e CRAL-FD apresentaram taxa média de entrega superior a 99%, este alto valor pode ser explicada pela escolha de rotas de alta qualidade (Figura 5). CT e SPT mostram queda na taxa de entrega quando o crescimento da rede é exponencial, isto é esclarecido pela opção de enlaces de baixa qualidade. O aprendizado deste resultado é a importância de se considerar enlaces com taxa de perda intermediária ao escolher rotas.



(a) Consumo de energia em Joules variando o número de nós sensores.

(b) Efeito *overhearing* variando o número de nós sensores.

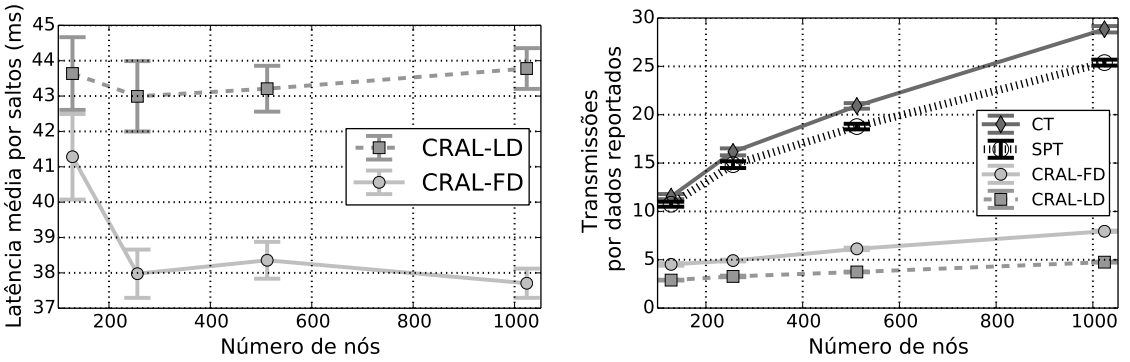
Figura 8. Avaliação do consumo de energia e efeito *overhearing* dos algoritmos.

Neste experimento é analisado o consumo de energia dos protocolos em termos de bateria e efeito *overhearing*. Na Figura 8(a) é apresentado o consumo em Joules dos protocolos, CRAL-LD e CRAL-FD se mostram mais econômicos do que SPT e CT. CRAL-FD supera o consumo de CRAL-LD, pois apresentar rotas com ETX médio ligeiramente superiores ao CRAL-LD (Figura 5). SPT e CT são similares para redes de porte pequeno e médio, porém CT apresenta maior consumo de energia em redes de grande porte, o que pode ser explicado pelo fato do uso do menor caminho em saltos até o sorvedouro. A Figura 8(b) exibe o *overhearing*[3] para os diferentes portes de rede. CRAL-LD é o algoritmo que apresenta menores valores, seguido por CRAL-FD e, por fim, CT. Estes resultados mostram o potencial do protocolo CRAL na economia de energia.

Para avaliar a eficiência (Seção 3) verificamos a latência[4] e o número de transmissões por dados gerados. A Figura 9(a) mostra a latência do CRAL-LD e CRAL-FD para diferentes portes da rede. CRAL-FD apresenta latência menor, pois escolhe rotas com maior vazão que CRAL-LD, visto que emprega o ETT como LQE. CT não é exibido, pois seus resultados apresentam valores, em média, 30% superiores aos aferidos por CRAL-LD ou CRAL-FD. A Figura 9(b) exibe o número de transmissões por dados

[3] *Overhearing*, neste trabalho, é definido como o número de mensagens escutadas pelo nó que não são destinadas ao mesmo acarretando em consumo de energia desnecessário.

[4] Neste trabalho, latência é o tempo médio por saltos das mensagens de dados.



(a) Latência média por saltos variando o número de nós sensores.

(b) Número de transmissões por dados reportados variando o número de nós sensores.

Figura 9. Avaliação eficiência dos algoritmos de roteamento em termos de latência por saltos e número de transmissões por dados reportados.

gerados. CRAL-LD e CRAL-FD necessitam, em média, 8 transmissões por mensagem de dados, ao passo que SPT e CT superam 25 transmissões. Alto número de transmissões por dado implica em gasto de energia, aumento do *overhearing* e alta latência.

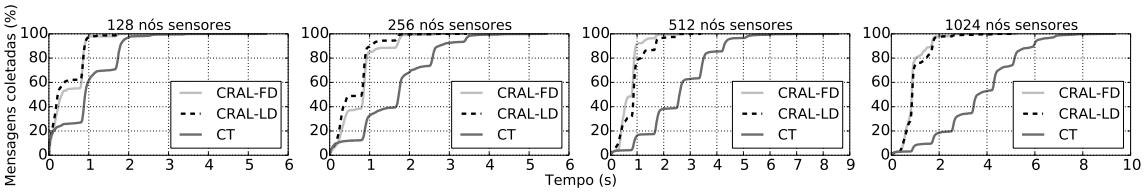
Para mostrar a flexibilidade do protocolo CRAL foram anexadas diferentes técnicas de fusão de dados (Seção 4.3). O desempenho dos algoritmos sobre o *freshness*[5] dos dados coletados é exibidos na Figura 10. A Figura 10(a) mostra a porcentagem de coletas e o tempo de entrega, usando*PerHopAd* e o sorvedouro no centro da rede (cenário que apresenta melhores resultados para todos protocolos). É fácil perceber o protocolo CRAL entrega rapidamente as mensagens. Em redes grandes, CRAL-LD e CRAL-FD entregam 90% ou mais das mensagens com apenas $2s$, enquanto CT no máximo 20% com o mesmo tempo. Esta característica é similar em todos os cenários avaliados. Nas Figuras 10(b)(c) são apresentados os resultados do *freshness*[6] para os algoritmos CRALs. Ao comparar as duas abordagens percebe-se pouca mudança. Assim, pode-se escolher CRAL-FD se o tempo de entrega de cada mensagem é um quesito mais importante ou CRAL-LD no caso da taxa de entrega ser o fator mais relevante.

Em relação à idade dos dados com e sem uso de funções de fusão de dados, percebe-se que as funções *Simple* e *PerHop* não são tão ágeis quanto *PerHopAd*, a explicação disto é o efeito cascata da função *PerHopAd*, o qual favorece a fusão de dados ao passo que mantém baixo o tempo de entrega dos dados. A combinação do *PerHopAd* com CRAL torna a fusão de dados ágil em redes de todos os portes experimentadas. O emprego de técnicas de fusão de dados com o CT diminui consideravelmente a velocidade da entrega dos dados, sendo pouco ágil e escalável.

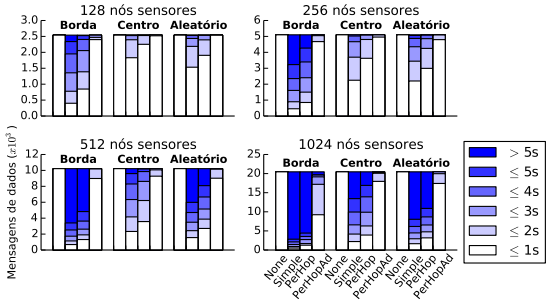
Um dos maiores benefícios das funções de fusão de dados é a redução da quantidade de transmissões na rede. Esta redução acarreta, por exemplo, em menor consumo de energia e atenuação do efeito *overhearing*. A Figura 11 mostra o número de transmissões algoritmos de roteamento em conjunto com as funções de fusão de dados. O compromisso entre transmissões e o *freshness* dos dados é analisado na Figura 11(a). CRAL-LD

[5]*Freshness* é o tempo que a mensagem foi gerada subtraído do tempo que foi coletada no sorvedouro.

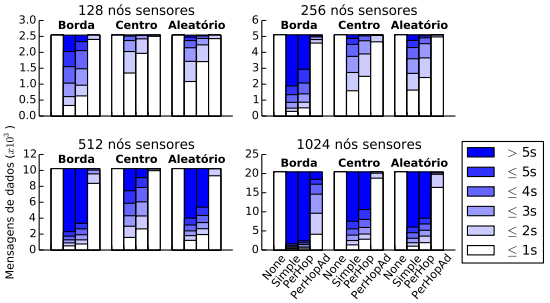
[6]Neste experimento, o *freshness* foi discretizado como $1s, 2s, ...$, assim para cada função as mensagens recebidas com até $1s, 2s, ...$ de atraso, o qual será chamado de idade da mensagem.



(a) Porcentagem de mensagens coletadas no tempo.



(b) *Freshness* para o algoritmo CRAL-LD.



(c) *Freshness* para o algoritmo CRAL-FD.

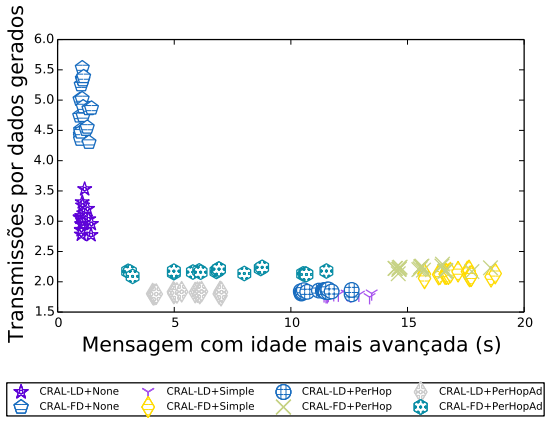
Figura 10. Avaliação do comportamento dos algoritmos de roteamento sobre a idade dos dados coletados em face às funções de fusão de dados.

e CRAL-FD sem técnicas de fusão apresentam baixo tempo de entrega dos dados, já quando se faz uso da fusão de dados tem-se maior tempo para entregar os dados porém com um número de transmissões maior. CRAL-FD em algumas simulações balanceia este compromisso obtendo baixo tempo de entrega e número de transmissões. Já CRAL-LD, reduz ao máximo o número de transmissões e mantém um tempo de entrega moderado entre $4s$ e $7s$. CT apresenta número de transmissões e tempo superiores ao do CRAL em todos os cenários avaliados. O número de transmissões por dados gerados para o da árvore CT é exibido na Figura 11(b). CT precisa de alto quantidade de transmissões para entregar dados, quando não são usadas funções de agregação. Já com o uso de funções de fusão de dados esse número é reduzido drasticamente, a maior diferença acontece quando a rede é de grande porte e o sorvedouro está na borda. Neste cenário, o uso das funções chega a transmitir 3 vezes menos do que o roteamento sem funções de fusão de dados.

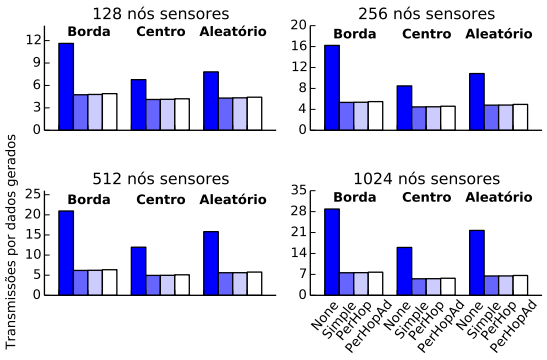
CRAL-LD e CRAL-FD se mostram mais econômicos do que CT em termos de transmissões por dados gerados. As Figuras 11(b)(c) mostram os resultados para respectivamente CRAL-LD e CRAL-FD, o número de transmissões dessas abordagens sem o uso de funções de fusão de dados é menor que CT com funções de fusão. Isto mostra que as abordagens são eficientes mesmo sem o uso de fusão de dados. Quando são empregadas as funções de fusão de dados, o número de transmissões cai pela metade na maioria dos casos, chegando a ser aproximadamente 3 vezes menor quando em redes grandes e sorvedouro posicionado na borda da rede com o uso do CRAL-FD. Os algoritmos CRAL, quando comparado com CT, apresentam substancial redução de transmissões e por consequência menor consumo de energia, tanto com e quanto sem o uso de fusão de dados.

6. Conclusão

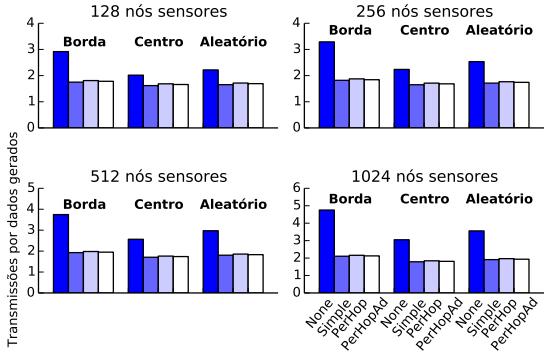
Neste artigo foi apresentado o protocolo CRAL que roteia mensagens baseado na importância topológica dos nós intermediários em RSSF. CRAL utiliza-se das métricas de



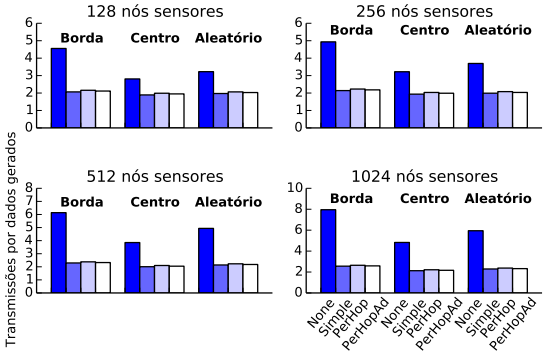
(a) Compromisso entre transmissões e *freshness*.



(b) Algoritmo CT.



(c) Algoritmo CRAL-LD.



(d) Algoritmo CRAL-FD.

Figura 11. Avaliação do número de transmissões por dados gerados para cada algoritmo de roteamento em face às funções de fusão de dados.

centralidade e estimadores de enlace para prover rotas de alta vazão, ao passo que economiza energia. CRAL é uma aproximação da árvore de Steiner, que é a solução ótima para o problema considerado no trabalho. Os resultados mostram que CRAL apresenta alta taxa de entrega alcançando até a 99.6%, reduz o número de transmissão por dados em até 3 vezes. CRAL-LD e CRAL-FD operando em conjunto com técnicas de fusão de dados, em especial com a técnica PerHopAd, se mostram flexíveis e adaptativos, permanecendo eficientes, pois mais de 90% das mensagens levam até $2s$ para serem entregues, enquanto CT entrega 20% dos dados no mesmo tempo. Além disso, CRAL reduz drasticamente o número de transmissões por dados gerados e consumo de energia na presença ou não de técnicas de fusão de dados. Estes resultados colocam CRAL como uma alternativa para roteamento de baixo consumo de energia e para fusão de dados em RSSF.

Os itens discutidos na Seção 4.4 já encontram-se em desenvolvimento, como o balanceamento do consumo de energia. Além disso, desenvolvemos um protocolo que permite confirmação de dados [Santos et al. 2015], assim será possível solucionar o problema da perda de pacotes agregados.

Referências

Almiron, M. G., Ramos, H., OLIVEIRA, E., AO GM DE MENEZES, J., Guidoni, D., Stancioli, P., da Cunha, F., de Aquino, A., Mini, R., Frery, A., et al. (2010). Redes complexas na modelagem de redes de computadores. *Minicurso SBRC*.

Baccour, N., Koubaa, A., Jamaa, M., Youssef, H., and Zuniga, M. (2009). A comparative simulation study of link quality estimators in wsn. In *MASCOTS '09*. IEEE.

Baccour, N., Koubâa, A., Mottola, L., Zúñiga, M. A., Youssef, H., Boano, C. A., and Alves, M. (2012). Radio link quality estimation in wireless sensor networks: A survey. *ACM Trans. Sen. Netw.*

Boukerche, A., Loureiro, A. A., Nakamura, E. F., Oliveira, H. A., Ramos, H. S., and Villas, L. A. (2014). Cloud-assisted computing for event-driven mobile services. *Mobile Networks and Applications*, 19(2):161–170.

De Couto, D. S. J., Aguayo, D., Bicket, J., and Morris, R. (2005). A high-throughput path metric for multi-hop wireless routing. *Wirel. Netw.*

Dolev, S., Elovici, Y., and Puzis, R. (2010). Routing betweenness centrality. *JACM*.

Draves, R., Padhye, J., and Zill, B. (2008). Routing in multi-radio, multi-hop wireless mesh networks. MobiCom, New York, USA.

Fasolo, E., Rossi, M., Widmer, J., and Zorzi, M. (2007). In-network aggregation techniques for wireless sensor networks: a survey. *Wireless Communications, IEEE*.

IEEE 802.11b (1999). `http://standards.ieee.org/`.

Javaid, N., Javaid, A., Khan, I., and Djouani, K. (2009). Performance study of etx based wireless routing metrics. In *2nd IC4 2009*.

Levis, P. A., Patel, N., Culler, D., and Shenker, S. (2003). *Trickle: A self regulating algorithm for code propagation and maintenance in wireless sensor networks*. Computer Science Division, University of California.

Li, J. and Mohapatra, P. (2005). An analytical model for the energy hole problem in many-to-one sensor networks. In *VTC-2005-Fall IEEE 62nd*.

Oliveira, E., Ramos, H., and Loureiro, A. (2010). Centrality-based routing for wireless sensor networks. In *Wireless Days, 2010 IFIP*.

Orinoco wireless networks (2014). `http://www.orinocowireless.com/`.

Ramos, H. S., Frery, A. C., Boukerche, A., Oliveira, E. M., and Loureiro, A. A. (2014). Topology-related metrics and applications for the design and operation of wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 10(3):53.

Santos, B. P., Vieira, M. A. M., and Vieira, L. F. M. (2015). extend collection tree protocol. In *2015 WCNC: Mobile and Wireless Networks*, pages 1530–1535, New Orleans, USA.

Sinalgo, D. C. G. (2014). Simulator for network algorithms (sinalgo).

Solis, I. and Obraczka, K. (2003). In-Network Aggregation Trade-offs for Data Collection in Wireless Sensor Networks.

Villas, L., Boukerche, A., de Araujo, R. B., and Loureiro, A. A. F. (2010). Highly dynamic routing protocol for data aggregation in sensor networks. In *ISCC 2010*. IEEE.

Alocação de Endereços IPv6 em Redes Multi-hop de Rádios de Baixa Potência

Bruna Soares Peres[1], **Olga Goussevskaia**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

{bperes,olga}@dcc.ufmg.br

Abstract. *In this work, we propose a Multi-hop Host Configuration strategy that explores cycle-free network structures in low-power wireless networks to generate and assign IPv6 addresses to nodes. The objective is to enable efficient and robust top-down routing with low memory footprint. MHCL generates and assigns IPv6 addresses which reflect the topology of the underlying wireless network. We implemented our strategy as a subroutine of RPL protocol in Contiki OS. We performed experiments to show that our strategy is efficient in time and number of messages, and is robust to network dynamics, caused by failures in network links and nodes.*

Resumo. *Muitas redes sem fio de baixa potência são baseadas em mecanismos que mantêm topologias acíclicas para suportar aplicações de coleta de dados, tipicamente otimizada para o tráfego ascendente de dados. Pacotes que precisam ser enviados em uma direção diferente devem seguir por caminhos mais longos e são frequentemente descartados devido à falta de memória para as tabelas de roteamento. Neste trabalho propomos uma estratégia de configuração de nós que explora estruturas acíclicas em redes sem fio de baixa potência para gerar e atribuir endereços IPv6 para os nós. O objetivo é permitir o roteamento descendente de maneira eficiente e robusta, com baixo consumo de memória. Isso é alcançado ao gerar e atribuir endereços IPv6 que refletem a topologia da rede sem fio. Nós implementamos a estratégia proposta como uma rotina do protocolo RPL no sistema operacional Contiki. Foram realizados experimentos para demostrar que a estratégia é eficiente em termos de tempo e número de mensagens, além de ser robusta à dinâmica da rede, decorrente de falhas em enlaces e nós da rede.*

1. Introdução

A principal função de uma rede sem fio de baixa potência é, tipicamente, um tipo de coleta de dados [Liu et al. 2013, Levis et al. 2008]. Aplicações baseadas na coleta de dados são muitas e os exemplos incluem monitoramento do meio ambiente [Tolle et al. 2005], segurança [Vicaire et al. 2009] e observações científicas [Werner-Allen et al. 2006]. De modo a realizar a coleta de dados, uma estrutura de grafo acíclico é tipicamente mantida e um *convergecast* (fluxo de dados de baixo para cima a partir das folhas em relação à raiz) é implementado nessa topologia de rede. Muitos sistemas operacionais para nós sensores (por exemplo, Tiny OS [Levis et al. 2005] e *Contiki OS* [Dunkels et al. 2004,

con]) implementam mecanismos para manter topologias de rede acíclicas para suportar aplicações de coleta de dados.

Em algumas situações, entretanto, o fluxo de dados na direção oposta torna-se necessário. Um algoritmo de coleta de dados muito comum em redes sem fio, conhecido como *convergecast*, pode ativar o tráfego de dados no sentido descendente em cenários onde falhas são comuns, por exemplo. O algoritmo de *convergecast* pode ser formalmente definidos como se segue: em um ambiente consistindo de nós de sensores sem fios, cada nó *i* possui uma unidade de dados que deve ser transmitida para a estação base. Cada nó possui um alcance de transmissão, que pode ser ajustado até um valor máximo de r. Todos os nós dentro da área de transmissão são considerados os seus vizinhos. Um nó sensor, se dentro da faixa, pode transmitir seus dados para a estação de base diretamente (*singlehop*) ou através de outro nó (*multihop*) [Upadhyayula et al. 2003]. Quando a rede opera em um cenário onde falhas são comuns, é possível que os nós optem por receber uma confirmação da raiz sobre o recebimento de suas mensagens, ou seja, a raiz ou roteador de borda deve enviar uma mensagem descendente para um determinado nó, assim que receber uma mensagem de coleta desse nó.

Mesmo o tráfego de dados de cima para baixo não sendo tipicamente o principal objetivo das redes sem fio *multi-hop* de baixa potência, ele é implementado por alguns esquemas de roteamento populares como uma função "extra", o que significa que as rotas são otimizadas para o tráfego de baixo para cima e mensagens descendentes podem seguir por rotas mais longas ou não serem entregues devido a restrições de memória. No protocolo RPL, por exemplo, é mantida uma topologia de rede acíclica, chamada DAG (*Directed Acyclic Graph*), em que cada nó mantém na memória uma pequena lista de nós pais, ordenados de acordo com a função objetivo, para os quais ele encaminha os dados em direção ascendente à raiz. Se um nó quer atuar como destino, ele envia uma mensagem para cima através do DAG, e os nós intermediários (se eles operam no assim chamado modo *storing*) adicionam uma entrada para este destino na tabela de roteamento criada especificamente para rotas descendentes. O tamanho de cada tabela deste tipo é potencialmente $O(n)$, onde n é o tamanho da sub-árvore com raiz em cada nó, ou seja, o número total de descendentes do nó. Dado que a capacidade de memória pode ser altamente restringida em redes sem fio de baixa potência, muitas vezes nem todas as rotas podem ser armazenadas e, consequentemente, pacotes serão descartados. Isto resulta na perda de um número elevado de mensagens e um alto consumo de memória. A fim de reduzir o tamanho da tabela de roteamento, seria desejável agregar vários endereços de nós de destinos próximos em uma única entrada da tabela.

Neste trabalho, propomos o MHCL (*IPv6 Multi-hop Host Configuration for Low-power wireless networks*)—uma estratégia de configuração de nós *multi-hop* que explora estruturas acíclicas em redes sem fio de baixa potência para gerar e atribuir endereços IPv6 para os nós. O objetivo é permitir tráfego descendente eficiente e robusto (broadcast, multicast e unicast) com baixo consumo de memória, ou seja, tabelas de roteamento pequenas. Propomos e analisamos duas estratégias de alocação de endereços: *top-down* e *bottom-up*. Na abordagem top-down, cada nó realiza um particionamento do espaço de endereços disponível em intervalos de endereços de tamanhos iguais e distribui essas fatias a seus filhos. A abordagem bottom-up contém uma fase inicial de agregação, em que nós computam seu número de descendentes. Uma vez que a raiz recebe o número

total de descendentes de cada um dos seus filhos (imediatos), atribui uma fatia/partição de endereços proporcional ao tamanho da sub-árvore com raiz em cada um de seus filhos. Uma vez que um nó recebe uma fatia de endereço de seu pai preferencial, ele particiona essa faixa entre seus próprios filhos, usando o mesmo algoritmo.

O MHCL gera e atribui endereços IPv6 que refletem a topologia da rede sem fio subjacente. Como resultado, as tabelas de roteamento para tráfego descendente são de tamanho $O(k)$, onde k é o número de filhos (imediatos) de cada nó de roteamento em uma topologia acíclica. Foram usados temporizadores inspirados no algoritmo *Trickle* [Levis et al. 2004] para ajustar o mecanismo de comunicação em nosso protocolo, de modo que ele é capaz de se adaptar rapidamente à dinâmica na topologia da rede, sem inundar a rede com mensagens de controle. Analisamos quão robusta é nossa estratégia em relação à dinâmica da rede, decorrente de falhas em enlaces e nós. Mostramos que, se os endereços são atribuídos na fase de configuração inicial da rede, as mensagens podem ser entregues com alta taxa de sucesso. Demonstramos por meio de simulações (usando o simulador *Cooja* [Eriksson et al. 2009] do sistema operacional *Contiki OS* [Dunkels et al. 2004] com implementação do protocolo RPL) que o MHCL é robusto à dinâmica da topologia de redes sem fio de baixa potência.

O restante deste artigo está organizado da seguinte forma. Na Seção 2, nós descrevemos brevemente como o tráfego descendente é implementado no protocolo RPL. Na Seção 3 apresentamos as versões top-down e bottom-up do MHCL. Na Seção 4 apresentamos nossos resultados experimentais. Na Seção 5 discutimos trabalhos relacionados e, na Seção 6 encerramos com algumas conclusões finais.

2. Sistemas 6LoWPAN

Sistemas 6LoWPAN (*IPv6 over Low power Wireless Personal Area Networks*[1]) são compostos de dispositivos de baixo custo que permitem comunicação sem fio, porém com recursos limitados, obedecendo ao padrão IEEE802.15.4. Nessas redes, perdas de pacotes são extremamente frequentes, e os links podem se tornar inutilizáveis por algum tempo [Vasseur and Dunkels 2010].

2.1. Endereçamento IPv6

Entre a rede (global) IPv6 e uma rede 6LoWPAN existe um roteador de borda que realiza a compressão do cabeçalho IPv6. Os 128 bits do endereço IPv6 são normalmente divididos em duas partes: o prefixo da rede (64 bits) e o endereço do nó (64 bits). O mecanismo de compressão de cabeçalho 6LoWPAN omite os bits do prefixo de rede, pois eles são fixos para uma determinada rede 6LoWPAN. Uma vez que os outros 64 bits podem abordar um espaço de endereçamento muito grande, o 6LoWPAN fornece opções para comprimir o endereço do nó (o uso comum é de 16 bits)[2]. Similarmente ao IPv4, existem duas maneiras de configuração de endereços: dinâmica ou estática. A atribuição de endereço estático precisa ser configurada manualmente em cada dispositivo. A configuração dinâmica é realizada pelo protocolo DHCPv6 (*Dynamic Host Configuration Protocol version 6*). Dado que a rede é *multihop*, isso requer a execução de um servidor DHCPv6 e vários agentes retransmissores na rede.

[1] `http://datatracker.ietf.org/doc/rfc4919`
[2] `http://datatracker.ietf.org/doc/rfc6282`

2.2. Protocolo RPL

O RPL [Winter et al. 2012] é um protocolo de roteamento especificamente projetado para redes 6LoWPAN. A base do protocolo é construção e manutenção de uma topologia acíclica de rede direcionada à raiz. Para isto, um Grafo Acíclico Direcionado Orientado ao Destino (DODAG) é criado. Para iniciar o processo de construção do DAG a raiz anuncia informações sobre o grafo por meio de mensagens do tipo DIO (*DODAG Information Object*). Uma vez que um nó se junta ao DAG, ele armazena uma lista de potenciais pais, ordenada pela qualidade das respectivas conexões, e marca o primeiro pai da lista como preferencial. Em seguida, o nó calcula seu *rank*, uma métrica que indica as coordenadas do nó na hierarquia da rede [Vasseur et al. 2011]. Quando esse processo se estabiliza, a coleta de dados pode começar.

RPL especifica dois modos de operação para o roteamento descendente: *storing* e *non-storing*. Ambos os modos requerem a geração de mensagens do tipo DAO (*Destination Advertisement Object*) pelos nós que desejam atuar como destinos, ou seja, receber mensagens endereçadas a eles. No modo de operação *storing*, cada roteador RPL devem armazenar rotas para seus destinos em um DODAG. Essas informações são repassadas dos nós para seus pais preferenciais. Dessa forma, a memória necessária em um nó próximo à raiz é $O(n)$, onde n é o tamanho da sub-árvore com raiz em cada nó, ou seja, o número total de descendentes do nó. Dado que a capacidade de memória pode ser altamente restringida em redes sem fio de baixa potência, muitas vezes nem todas as rotas podem ser armazenadas e, consequentemente, pacotes serão descartados. Isso resulta na perda de um número elevado de mensagens e um alto consumo de memória. No modo *non-storing*, os nós não possuam capacidade de armazenamento e as mensagens DAO são enviadas diretamente para a raiz. A raiz é o único nó que armazena a tabela de roteamento com as rotas descendentes. Para enviar mensagens para os nós, é então utilizado o roteamento de origem (*source routing*), ou seja, as rotas são enviadas dentro dos próprios pacotes.

2.3. Trickle

O algoritmo *Trickle* permite que nós em um meio compartilhado com perdas possam trocar informações de uma maneira altamente robusta, simples e escalável, com um baixo consumo de energia [P. Levis 2011]. Ao ajustar as janelas de transmissão dinamicamente, o Trickle é capaz de difundir novas informações sobre a escala de tempo de transmissão da camada de enlace enquanto envia apenas algumas mensagens por hora quando a informação não se altera. No algoritmo *Trickle*, esporadicamente um nó transmite dados sobre a topologia, a não ser que ele escute alguma outra transmissão com dados que sugiram que sua própria transmissão é redundante [Levis et al. 2004]. Existem dois resultados possíveis de uma mensagem do Trickle. Cada nó que escuta a mensagem pode perceber que os dados da mensagem são consistentes com seu próprio estado ou um receptor pode detectar uma inconsistência. Dessa forma, o algoritmo Trickle pode ser utilizado para para o estabelecimento de eventual consistência de uma rede sem fio e para otimizar a disseminação de informações sobre a topologia, como é o caso de sua utilização no protocolo RPL [Winter et al. 2012]. No MHCL, o Trickle foi utilizado como base para a definição de um temporizador responsável pela estabilização da rede. Quando uma informação sobre a topologia não se altera, o temporizador tem seu valor dobrado até atingir o máximo. Caso a informação se altere, o valor é redefinido ao inicial e o nó

deve esperar até o temporizador chegar ao valor máximo para ter certeza que a topologia está estável.

3. MHCL: Alocação de Endereços IPv6 em Redes Multi-hop de Rádios de Baixa Potência

O objetivo do MHCL (*IPv6 Multi-hop Host Configuration in Low-power wireless networks*) é implementar um esquema de alocação de endereços IPv6 para tráfego descendentes que consuma pouca memória, ou seja, necessite de pequenas tabelas de roteamento. O espaço de endereços disponível do roteador de borda, os 64 bits menos significativos do endereço IPv6, é particionado de maneira hierárquica entre os nós conectados por meio de uma topologia *multihop* acíclica. Cada nó recebe uma faixa de endereços IPv6 de seu pai e particiona essa faixa entre seus filhos, até que todos os nós tenham sido endereçados. Uma vez que a alocação de endereços é realizada de maneira hierárquica, a tabela de roteamento de cada nó deve ter k entradas, onde k é o número de filhos (diretos). Cada entrada da tabela de roteamento agrega o endereço de todos os nós na subárvore com raiz no nó filho correspondente.

Para decidir como o espaço de endereçamento disponível deve ser particionado, os nós precisam coletar informações sobre a topologia da rede. Uma vez que uma versão *estável* da topologia da rede é alcançada, a raiz começa a distribuir as fatias de endereços. Note que o conceito de estabilidade é importante para implementar um particionamento de endereços coerente. Portanto, o MHCL possui uma fase de inicialização, durante a qual informações sobre a topologia são atualizadas progressivamente, até que um período de tempo (pré-definido) suficientemente longo se passe sem alterações na topologia. Note que, uma vez que a rede atinge um estado de estabilidade inicial, mudanças de topologia futuras são esperadas ser de natureza local, provocadas por uma conexão falha ou falha de um nó. No (atípico) caso de mudanças globais e permanentes na topologia da rede, o algoritmo MHCL deve ser reiniciado pela raiz.

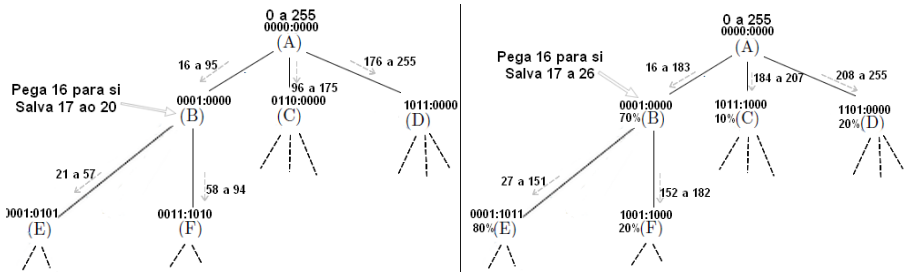
No restante desta seção nós descrevemos os principais componentes do MHCL: a alocação de endereços IPv6 (Seção 3.1), os tipos de mensagens (Seção 3.2), as rotinas de comunicação e temporizadores (Seção 3.3) e as tabelas de roteamento e rotinas de encaminhamento (Seção 3.4).

3.1. Alocação de Endereços IPv6

Nós propomos duas estratégias de particionamento do espaço de endereçamento: top-down e bottom-up. Na abordagem top-down (Figura 1(a)), cada nó conta o seu número k de filhos, espera até que seu pai preferencial lhe aloque uma faixa de endereços (caso o nó não seja raiz), particiona o seu espaço de endereçamento em k faixas de igual tamanho, reservando uma parte de $r\%$ do total para possíveis conexões futuras, e então distribui as faixas resultantes para os filhos. Note que esta abordagem utiliza apenas informação sobre nós a uma distância de um *hop*, o que lhe proporciona uma fase relativamente rápida de inicialização.

Na abordagem bottom-up (Figura 1(b)), cada nó contabiliza, através de uma etapa de agregação, o número total de seus descendentes e o repassa para o seu pai preferencial. Além disso, cada nó mantém o número de descendentes de cada filho. Uma vez que as informações se estabilizam, a raiz inicia o processo de distribuição de endereços. Para

isso, ela particiona o espaço de endereçamento disponível entre os seus filhos, de forma que cada um receba uma faixa de tamanho proporcional ao seu respectivo número de descendentes, deixando novamente uma faixa de endereços de reserva. Cada nó, ao receber a sua faixa do pai, repete o procedimento de particionamento proporcional ao número de descendentes de cada filho, até que todos os nós sejam endereçados. A ideia é alocar faixas de endereços maiores para subárvores maiores, de forma a maximizar o aproveitamento do espaço de endereçamento. Isso se torna especialmente relevante em redes muito grandes. Note que esta abordagem utiliza informação agregada ao longo de vários *hops*, o que torna a sua fase de inicialização relativamente mais longa.



(a) Top-down (b) Bottom-up

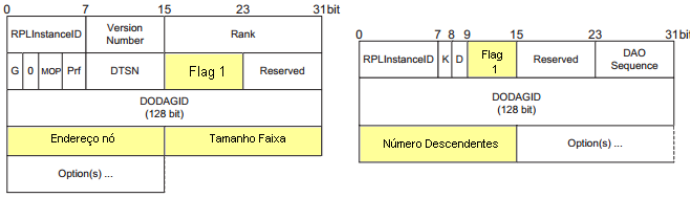
Figura 1. Exemplos de particionamento do espaço de endereçamento.

3.2. Mensagens

MHCL utiliza quatro tipos de mensagens, implementadas modificando-se as mensagens DIO e DAO do RPL: DIO_{MHCL}, $DIOACK_{MHCL}$, DAO_{MHCL} e $DAOACK_{MHCL}$. Mensagens do tipo DIO_{MHCL} (Figura 2(a)) são enviadas em rotas descendentes, os seja, dos pais para os filhos. Essa mensagem é utilizada durante a fase do endereçamento, pois ela carrega as informações necessárias para um nó definir seu endereço e a faixa recebida de seu pai. O campo *flag* da mensagem DIO do RPL é definido em 1 e o endereço do nó e tamanho da faixa de endereços são enviados no campo *options*. Mensagens do tipo $DIOACK_{MHCL}$ são identificadas por $flag = 2$ e usadas para confirmar o recebimento de mensagens DIO_{MHCL}. Mensagens do tipo DAO_{MHCL} (Figura 2(b)) são enviadas em sentido ascendente, de filho para pai, e possuem duas funções: na abordagem bottom-up, elas carregam o número de descendentes do nó e na abordagem top-down, servem para informar ao pai que ele foi escolhido como preferencial pelo respectivo filho. Essas mensagens são confirmadas por mensagens do tipo $DAOACK_{MHCL}$.

3.3. Comunicação e Temporizadores

O algoritmo MHCL top-down começa com a rotina de *informar pai preferencial*. Se um nó não é raiz, ele deve informar a seu pai que ele é um de seus filhos, enviando uma mensagem DAO_{MHCL}. Um nó deve esperar sua lista de potenciais pais se estabilizar e só então informar ao pai escolhido. No Algoritmo 1, duas constantes são usadas: DIO_{min} é um parâmetro de temporização usado pelo RPL para que um nó defina o menor intervalo para enviar mensagens DIO. $peFilho$ é um *parâmetro de estabilização* usado para decidir quando o pai preferido está estabilizado (a Tabela 1 contém os valores desses parâmetros usados nas simulações). Uma vez que a escolha do pai está estável, a variável local $definiuPai$ é definida em *TRUE* e uma mensagem DAO_{MHCL} é enviada para o pai



(a) DIO_{MHCL} (b) DAO_{MHCL}

Figura 2. Tipos de mensagens usados no MHCL.

para informar sobre a decisão (linhas 12–18, Algoritmo 1). Em seguida, cada nó ao receber mensagens (DAO_{MHCL}) de seus filhos, salvar uma lista com os filhos[3], atualiza o contador de filhos e confirma o recebimento do pacote para cada filho, enviando um $DAOACK_{MHCL}$. O processo de contagem dura até a informação se tornar estável, o que é controlado pelo parâmetro $pePai$. Uma vez que o número de filhos se torna estável, a variável local $definiuFilhos$ se torna verdadeira (linha 13, Algoritmo 2).

Na fase de endereçamento, se um nó não é raiz, ele deve esperar até receber uma faixa de endereços alocada a ele por seu pai (em uma mensagem DIO_{MHCL}) e confirmar o recebimento enviando uma mensagem $DIOACK_{MHCL}$. O particionamento e alocação das faixas de endereços é iniciado pela raiz assim que ela determina o número de filhos que possui ($definiuFilhos$ é verdadeiro). A faixa de endereços disponível é particionada em porções iguais entre os filhos conhecidos, guardando uma reserva para futuros filhos, ou conexões atrasadas. Uma vez que o endereço é particionado, a fatia correspondente a cada filho é enviada em uma mensagem DIO_{MHCL}, como mostra o Algoritmo 5. Se uma mensagem DAO_{MHCL} é recebida de um novo filho após a etapa de distribuição de endereços ter sido completada, então o processo de partição e alocação de endereços é repetido, usando a faixa reserva (a faixa reserva é a quantidade de endereços que o nó possui após realizar o endereçamento de seus filhos).

```
1:  definiuPai = FALSE;
2:  maxTimer = peFilho * DIO_min ;
3:  timer = rand(1/2 * DIO_min, DIO_min];
4:  while NÃO definiuPai do
5:      if NÃO-RAIZ and ESTOUROU TIMER then
6:          if MUDOU-PAI then
7:              reseta timer;
8:          else
9:              if timer < maxTimeR then
10:                 timer *= 2;
11:             else
12:                 definiuPai = TRUE;
13:                 if MHCL-top-down then
14:                     envia DAO_MHCL para pai;
15:                     if NÃO DAOACK_MHCL then
16:                         envia DAO_MHCL para pai;
17:                     end if
18:                 end if
19:             end if
20:         end if
21:     end if
22: end while
```

Algoritmo 1: MHCL: Timer Pai Preferido

```
1:  definiuFilhos = FALSE;
2:  maxTimer = pePai * DIO_min;
3:  timer = rand(1/2 * DIO_min, DIO_min];
4:  count = 0;
5:  while NÃO definiuFilhos do
6:      if ESTOUROU-TIMER then
7:          if MUDOU-CONTADOR then
8:              reseta timer;
9:          else
10:             if timer < maxTimer then
11:                 timer *= 2;
12:             else
13:                 definiuFilhos = TRUE;
14:             end if
15:         end if
16:     end if
17: end while
```

Algoritmo 2: MHCL top-down: Timer Contador Filhos

[3]Note que na implementação padrão de estruturas acíclicas em protocolos como RPL e CTP, um nó não salva uma lista de filhos, apenas uma lista de potenciais pais

A abordagem Bottom-Up do MHCL primeiramente executa um procedimento de agregação para computar o número total de descendentes de cada nó. Se um nó não é a raiz e ele já definiu seu pai preferido ($definiuPai$ é verdadeiro) ele começa enviando uma mensagem DAO_{MHCL} com o contador $count = 0$ (Algoritmo 3). O nó então, após enviar seu primeiro DAO_{MHCL} espera por mensagens desse tipo de seus filhos, atualizando o número de descendentes a cada mensagem recebida e controlando o temporizador de estabilização. Se um nó é a raiz, então ele só atualiza seu número de descendentes até seu número total estar estável (Algoritmo 4). Os parâmetros $peFolha$ e $peRaiz$ são usados para definir o critério de estabilização em nós não raiz e na raiz, respectivamente. Uma vez que a fase de agregação está completa, a variável local da raiz $definiuDescendentes$ é definida em verdadeiro, e o processo de alocação de endereços é iniciado pela raiz (Algoritmo 5).

```
1:  maxTimerFolha = peFolha * DIO_min;
2:  timer = rand(1/2 * DIO_min, DIO_min];
3:  count = 0;
4:  while NÃO-DIO_MHCL-DE-PAI do
5:      if NÃO-RAIZ and ESTOUROU-TIMER then
6:          if definiuPai and (count < 1) then
7:              envia DAO_MHCL para pai;
8:          end if
9:          if MUDOU-COUNT then
10:             envia DAO_MHCL para pai;
11:             reseta timer;
12:         else
13:             if timer < maxTimerFolha then
14:                 timer *= 2;
15:             end if
16:         end if
17:     end if
18: end while
```

Algoritmo 3: MHCL bottom-up: Timer Agregação

```
1:  definiuDescendentes = FALSE;
2:  maxTimerRaiz= peRaiz * DIO_min;
3:  timer = rand(1/2 * DIO_min, DIO_min];
4:  count = 0;
5:  while NÃO definiuDescendentes do
6:      if É-RAIZ and ESTOUROU-TIMER then
7:          if MUDOU-COUNT then
8:              reseta timer;
9:          else
10:             if timer < maxTimerRaiz then
11:                 timer* = 2;
12:             else
13:                 definiuDescendentes = TRUE;
14:             end if
15:         end if
16:     end if
17: end while
```

Algoritmo 4: MHCL bottom-up: Agregação (Raiz)

3.4. Tabelas de Roteamento e Encaminhamento

Cada nó, após o novo endereçamento, deve possuir uma tabela de roteamento com uma entrada para cada filho. Cada entrada possui duas informações: o endereço final da faixa de endereços do filho, e o endereço do filho. Dessa forma, para realizar o encaminhamento de mensagens basta comparar o endereço do destino com o último endereço da faixa de cada filho (Algoritmo 6). Como a inserção na tabela de roteamento é realizada de forma sequencial, os endereços estão em ordem crescente na tabela.

```
1:  ESTÁVEL = FALSE;
2:  if MHCL-top-down then
3:      ESTÁVEL = definiuFilhos;
4:  else
5:      ESTÁVEL = definiuDescendentes or NÃO-RAIZ;
6:  end if
7:  if ESTÁVEL and (É-RAIZ or RECEBEU-DIO_MHCL)
    then
8:      divide endereços disponíveis;
9:      for cada filho c_i do
10:         envia DIO_MHCL para c_i;
11:         if NÃO DIOACK_MHCL then
12:             envia DIO_MHCL para c_i;
13:         end if
14:     end for
15: end if
```

Algoritmo 5: MHCL: Distribuição de endereços IPv6

```
1:  i = 0;
2:  while IPv6-destino > finalFaixaIP[i] do
3:      i++;
4:  end while
5:  encaminha para filhos[i];
```

Algoritmo 6: Encaminhamento de Mensagens

4. Resultados de Simulações

O protocolo MHCL apresenta características compatíveis com as necessidades de uma rede sem fio de baixa potência. Preparado para possuir um comportamento robusto em caso de falhas e com um algoritmo adaptado para as restrições da rede, ele se mostra uma boa alternativa para o endereçamento de nós em redes IPv6 *multihop*, auxiliando no roteamento e solucionando problemas em protocolos existentes. A fim de comprovar esses resultados, simulações foram realizadas.

Na Seção 4.1 são descritos os cenários e parâmetros de configuração das simulações. Na Seção 4.2 a fase inicialização da rede é avaliada de acordo com o tempo, número de mensagens enviadas e número de nós endereçados. Por fim, na Seção 4.3 são avaliadas as taxas de entrega das mensagens de aplicação.

4.1. Configurações da Simulação

O algoritmo MHCL foi implementado como uma rotina do protocolo RPL no sistema operacional *Contiki* [Dunkels et al. 2004]. Os resultados apresentam uma comparação entre o protocolo RPL com sua implementação padrão no *Contiki OS*, que utiliza uma configuração estática de endereços IPv6. Para realizar os experimentos, foi utilizado o Cooja [Eriksson et al. 2009], um simulador de redes sem fio de baixa potência com nós do tipo *SkyMote*, com 10KB de memória RAM, executando o sistema operacional *Contiki OS* com implementação do protocolo RPL [Dunkels et al. 2004].

Foram simuladas redes com 9, 25, 49 e 81 nós em uma distribuição regular, com raio de comunicação de 50 metros e 35 metros de distância entre cada nó. Para cada valor apresentado anteriormente, foram realizadas 8 simulações. Além disso, foram considerados modelos com e sem falhas na rede. As falhas simuladas foram de dois tipos: TX e RX. Segundo implementação do *Cooja*, em falhas do tipo TX a falha está na transmissão de pacotes, sendo que em $1 - TX\%$ de transmissões, nenhum dispositivo recebe o pacote enviado. Em falhas do tipo RX, em $1 - RX\%$ de transmissões de pacotes, apenas o nó destino não recebe o pacote, ou seja, é como se a falha ocorresse no link. Nos nossos experimentos, avaliamos a robustez do protocolo MHCL a cada tipo de falha separadamente. Portanto, foram aplicadas falhas com taxa de 95% não simultaneamente, ou seja, quando uma das variáveis está em 95% a outra, obrigatoriamente, está em 100%.

Na camada de aplicação, nós utilizamos como base um exemplo disponível no *Cooja* (rpl-udp), que utiliza o protocolo UDP na camada de transporte, ou seja, não implementa confirmação ou retransmissão de pacotes. Fizemos algumas modificações na aplicação, de forma que cada nó envie uma mensagem para a raiz e ela responda a cada uma dessas mensagens. Dessa forma, a aplicação envia $n - 1$ mensagens ascendentes e $n - 1$ descendentes. Essas mensagens começam a ser trocadas após 180 segundos de simulação. A simulação termina quando a última mensagem de aplicação é respondida, ou quando o tempo para esse evento ser completado se encerra, em caso de falhas. A Tabela 1 apresenta os parâmetros dos algoritmos apresentados na Seção 3, que são usados para estimar o tempo de estabilização da rede da seguinte forma: cada temporizador usado no protocolo MHCL começa com um valor mínimo, igual a $I = 2^{DIO_{min}}$ ms (um parâmetro usado pelo próprio RPL) e um valor máximo, igual a $pe * 2^{DIO_{min}}$, onde pe é o parâmetro de estabilização de cada rotina executada (veja os algoritmos apresentados na Seção 3).

Parâmetro	peFilho	pePai	peFolha	peRaiz	DIO_{min}
Valor	2	4	4	8	6

Tabela 1. Valor dos parâmetros de estabilização nos algoritmos

4.2. Inicialização da Rede

Para avaliar a inicialização da rede três parâmetros foram considerados. O primeiro deles, o tempo de inicialização, leva em consideração quanto tempo a rede precisa para se estruturar e obter as informações necessárias para realizar o roteamento. Os dois outros aspectos avaliados são a quantidade de mensagens do tipo DAO e DIO trocadas durante essa fase de inicialização.

4.2.1. Número de Mensagens

Para avaliar a quantidade de mensagens enviadas por cada protocolo, os números de mensagens DAO e DIO foram contados separadamente. A Figura 4 apresenta os resultados obtidos. Como esperado, o número de DIOs enviados não se difere estatisticamente entre os algoritmos, com uma confiança de 95% (Figura 3(b)). Isto porque o mecanismo de descoberta e manutenção do DAG não foi alterado da implementação do RPL para o MHCL. Para o endereçamento dos nós no MHCL são utilizadas mensagens do tipo DIO, entretanto o número de mensagens adicionas é muito pequeno se comparado com a quantidade total de mensagens DIO transmitidas na rede. Entretanto, como o mecanismo de descoberta de rotas, que utiliza mensagens DAO, é diferente no RPL e MHCL, o número de mensagens desse tipo enviadas tem uma diferença significativa (Figura 3(a)). Com 95% de confiança podemos afirmar que o MHCL envia menos mensagens do tipo DAO do que o RPL, em média $10\times$ menos mensagens. Isto porque no MHCL existe um algoritmo de coleta de informações sobre a topologia que, uma vez encerrado, não envia mais mensagens DAO. O RPL realiza a notificação de rotas durante o funcionamento da rede, o que faz com que mensagens DAO sejam transmitidas continuamente para notificação de novas rotas.

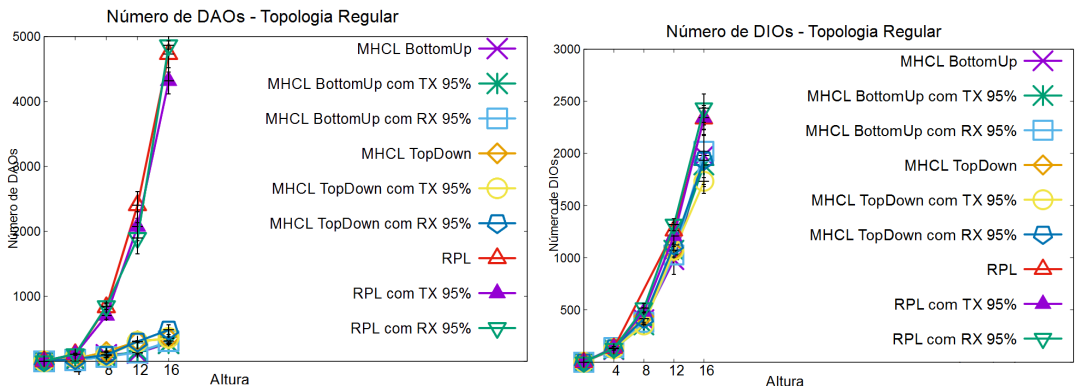
4.2.2. Tempo de Inicialização

Para o algoritmo MHCL foi considerado como tempo de inicialização o tempo necessário para todos os nós serem endereçados. No RPL foi medido o tempo decorrido até a raiz ter conhecimento de $n - 1$ rotas, ou seja, conhecer uma rota para cada nó da rede. A Figura 4(a) apresenta o gráfico com o tempo de inicialização de cada protocolo. Podemos observar que o tempo de inicialização tem um crescimento linear com o número de nós da rede, para todos os algoritmos simulados. No entanto, se considerarmos um intervalo de confiança de 95%, não é possível afirmar que um algoritmo é mais rápido do que o outro, pois existe a sobreposição dos intervalos de confiança. Note que estamos trabalhando com algoritmos assíncronos e a inicialização de cada nó é aleatória, o que é capaz de alterar significativamente o tempo de execução. Para isto, foi realizado o *Teste t.* Na comparação das estratégias bottom-up e top-down o intervalo de confiança resultante foi (-44.578,74.442), para a comparação entre o RPL e o top-down (-66.81,41.21) e, por fim, entre o RPL e a estratégia bottom-up (-97.022,41.578). Com esses resultados podemos

concluir que com 95% de confiança, o tempo dos três algoritmos é estatisticamente igual, uma vez que os intervalos de confiança incluem o valor zero, para 81 nós.

4.2.3. Taxa de Endereçamento

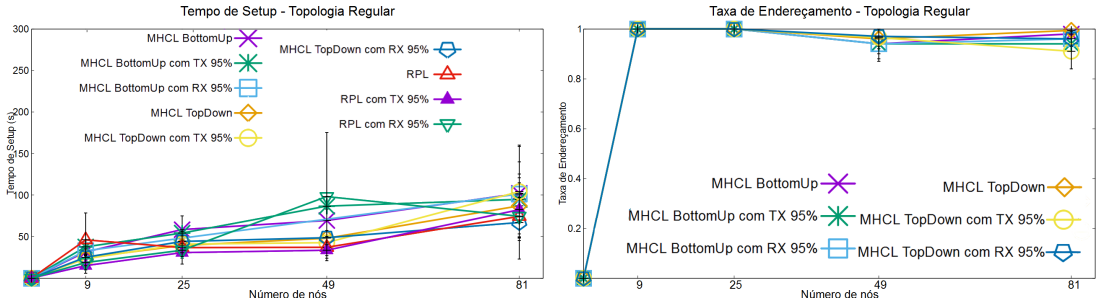
O endereçamento dos nós pelo MHCL é feito durante a inicialização da rede. Como a rede pode apresentar colisões e falhas nos nós e nos links, algumas mensagens de endereçamento (do tipo DIO_{MHCL}) podem ser perdidas, mesmo usando mensagens de confirmação (do tipo $DIOACK_{MHCL}$), e realizando tentativas de retransmissão (nas simulações apresentadas, até 3 vezes). A Figura 4(b) apresenta os gráficos que compara a taxa de endereçamento de nós das estratégias bottom-up e top-down. Podemos observar que, apesar de colisões e falhas intermitentes de nós e links, a taxa de endereçamento se mantém entre 90% e 100% em ambas as estratégias. Como é possível observar na Figura 4(b), existe a sobreposição dos intervalos de confiança e ambos contêm a média da outra estratégia. Portanto, podemos afirmar que, estatisticamente, o endereçamento de ambas as estratégias apresenta uma mesma taxa de sucesso.



(a) Número de DAOs em função da altura do DAG.

(b) Número de DIOs em função da altura do DAG.

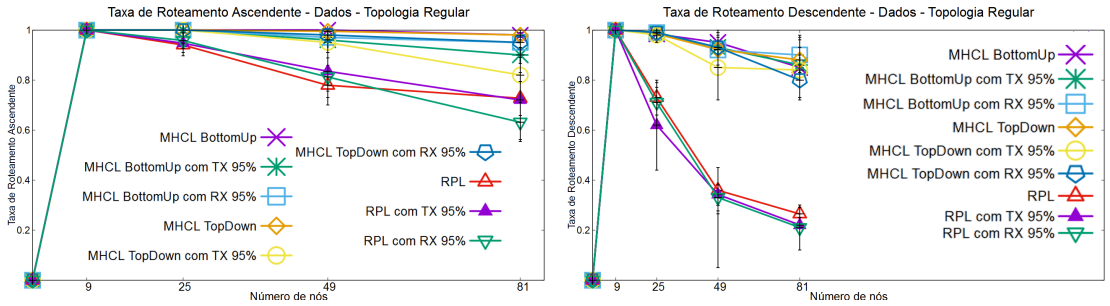
Figura 3. Número de mensagens na inicialização da rede.



(a) Tempo de incialização da rede.

(b) Taxa de endereçamento MHCL bottom-up e top-down.

Figura 4. Tempo de incialização e taxa de endereçamento.



(a) Taxa de entrega de mensagens de aplicação ascendentes.

(b) Taxa de entrega de mensagens de aplicação descendentes.

Figura 5. Taxa de entrega de mensagens de aplicação.

4.3. Mensagens de Aplicação

Por fim, a taxa de entrega de mensagens ascendentes e descendentes foi calculada (Figura 5). Note que a camada de aplicação simulada não realiza confirmações e retransmissões de mensagens, portanto, uma colisão ou falha em um link ou nó pode causar perdas definitivas de mensagens. Para o caso de mensagens ascendentes (Figura 5(a)), como o algoritmo de encaminhamento não foi alterado, as estratégias apresentam valores próximos. Entretanto, como durante o algoritmo de aplicação o RPL ainda envia mensagens do tipo DAO, essas mensagens disputam o meio com as mensagens ascendentes da aplicação, aumentando chance de colisões. No MHCL a transmissão de DAO após a fase de inicialização não ocorre, resultando em taxa de sucesso maior.

Para o roteamento descendente (Figura 5(b)) a diferença de desempenho entre MHCL e RPL é visível. Como é possível observar, enquanto o RPL, para 81 nós, apresenta uma taxa de entrega de 26% em média, o MHCL bottom-up e top-down obtêm sucesso em cerca de 87% das mensagens descendentes. Com 95% de confiança, podemos afirmar que o MHCL possui uma taxa de entrega de mensagens descendentes $3\times$ superior ao RPL, e as estratégias bottom-up e top-down do MHCL são equivalentes, em relação a essa taxa. Isto ocorre pois, no RPL, os nós não têm memória suficiente para armazenar as tabelas de roteamento necessárias para endereçar todos os seus descendentes, impossibilitando assim o roteamento de todas as mensagens, levando a uma baixa taxa de roteamento descendente.

5. Trabalhos Relacionados

O protocolo RPL, desde sua criação, foi muito estudado e, por isso, existem diversos trabalhos avaliando sua especificação e funcionamento. Em [Clausen et al. 2011] é realizada uma avaliação crítica do RPL, cerca de dois anos após sua criação. A falta de especificação de algumas partes do protocolo é um problema apontado, que conta com diversos pontos ainda não bem definidos do protocolo. Enquanto as mensagens do tipo DIO utilizam o Trickle para especificar a temporização de mensagens de maneira simples e de fácil entendimento, detalhes sobre mensagens do tipo DAO são obscuros e nenhum temporizador para essas mensagens é definido. Em [Ko et al. 2011] e [Zhang and Li 2014] o protocolo RPL é avaliado de acordo com sua implementação em cada um dos sistemas operacionais. No primeiro trabalho, [Ko et al. 2011], é apresentado uma implementação do RPL, chamada *TinyRPL*. Em seu trabalho, eles usam a pilha IP de baixa potência de

Berkley (BLIP) no *TinyOs* [Levis et al. 2005], que interage com sua implementação do protocolo. A implementação do *TinyRPL* conta com todas os mecanismos básicos da definição do RPL, enquanto omite todos os opcionais. Para a validação, o protocolo RPL é comparado com o *Collection Tree Protocol* (CTP) [Gnawali et al. 2009], o protocolo padrão para coleta de dados do *TinyOs*. Um dos benefícios do RPL, em comparação com o CTP, é a possibilidade de tipos variados de padrão de tráfego (P2P, MP2P e P2MP), além de sua capacidade de conectar nós à Internet diretamente, trocando pacotes com endereços IPv6 global. Em [Zhang and Li 2014] é estudada a performance do protocolo RPL no sistema operacional Contiki OS [Dunkels et al. 2004]. O algoritmo é simulado a fim de mostrar na prática implicações baseadas na descrição do protocolo. Por exemplo, de acordo com o mecanismo de inicialização da rede, é possível imaginar que mensagens de controle podem inundar a rede durante a construção do DAG e que essa etapa pode levar muito tempo. Os resultados mostram que inicialmente o número de mensagens de controle é superior ao número de mensagem de dados, mas essas mensagens tendem a diminuir após cerca de 60 segundos, o que pode ser determinado como o tempo de inicialização da rede, para os casos avaliados.

6. Conclusões

Neste trabalho nós propusemos o MHCL, uma estratégia de alocação de endereços IPv6 para redes sem fio *multihop* de baixa potência. As principais características do MHCL são: (1) Eficiência de memória: ao utilizar a topologia acíclica da rede para realizar um particionamento hierárquico do espaço de endereçamento disponível para um roteador de borda, os endereços de nós pertencentes a uma mesma sub-árvore são agrupados em uma única entrada na tabela de roteamento de um nó, tornando o tamanho da tabela $O(k)$, onde k é o número de filhos diretos do nó; isso se contrapõe ao tamanho $O(n)$ das tabelas de roteamento do RPL, onde n é o número total de descendentes de cada nó; (2) Eficiência temporal: as principais rotinas do MHCL são baseadas em temporizadores que rapidamente se adaptam à dinâmica da rede, possibilitando uma rápida fase de configuração de nós; (3) Eficiência em número de mensagens: o número de mensagens enviadas é adaptado à dinâmica da rede, através dos mesmos temporizadores acima; (4) Eficiência de endereçamento: o MHCL consegue endereçar perto de 100% dos nós, mesmo em cenários com falha; (5) Eficiência na entrega de mensagens de aplicação: o roteamento descendente do MHCL se mostrou eficaz, entregando até $3\times$ mais mensagens do que o RPL. Como trabalhos futuros, pretendemos analisar a robustez do MHCL a mudanças na topologia da rede que se estendam além de falhas intermitentes de nós e links individuais.

Referências

Contiki: The open source os for the internet of things. `http://www.contiki-os.org/`.

Clausen, T., Herberg, U., and Philipp, M. (2011). A critical evaluation of the ipv6 routing protocol for low power and lossy networks (rpl). In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*, pages 365–372.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, LCN '04, pages 455–462, Washington, DC, USA. IEEE Computer Society.

Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., and Marrón, P. J. (2009). Cooja/mspsim: Interoperability testing for wireless sensor networks. In *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 27:1–27:7, ICST, Brussels,

Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

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, New York, NY, USA. ACM.

Ko, J., Dawson-Haggerty, S., Gnawali, O., Culler, D., and Terzis, A. (2011). Evaluating the Performance of RPL and 6LoWPAN in TinyOS. In *Proceedings of Extending the Internet to Low power and Lossy Networks (IPSN 2011)*.

Levis, P., Brewer, E., Culler, D., Gay, D., Madden, S., Patel, N., Polastre, J., Shenker, S., Szewczyk, R., and Woo, A. (2008). The emergence of a networking primitive in wireless sensor networks. *Commun. ACM*, 51(7):99–106.

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D. (2005). TinyOS: An Operating System for Sensor Networks. In Weber, W., Rabaey, J., and Aarts, E., editors, *Ambient Intelligence*, chapter 7, pages 115–148. Springer Berlin Heidelberg, Berlin/Heidelberg.

Levis, P., Patel, N., Culler, D., and Shenker, S. (2004). Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, pages 2–2, Berkeley, CA, USA. USENIX Association.

Liu, Y., He, Y., Li, M., Wang, J., Liu, K., and Li, X.-Y. (2013). Does wireless sensor network scale? a measurement study on greenorbs. *IEEE Trans. Parallel Distrib. Syst.*, 24(10):1983–1993.

P. Levis, T. Clausen, J. H. O. G. J. K. (2011). The trickle algorithm. `https://tools.ietf.org/html/draft-ietf-roll-trickle-08`.

Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D., and Hong, W. (2005). A macroscope in the redwoods. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05, pages 51–63, New York, NY, USA. ACM.

Upadhyayula, S., Annamalai, V., and Gupta, S. (2003). A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 6, pages 3525–3530 vol.6.

Vasseur, J., Agarwal, N., Hui, J., Shelby, Z., Bertrand, P., and Chauvenet, C. (2011). Rpl: The ip routing protocol designed for low power and lossy networks. Internet Protocol for Smart Objects (IPSO) Alliance.

Vasseur, J.-P. and Dunkels, A. (2010). *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Vicaire, P., He, T., Cao, Q., Yan, T., Zhou, G., Gu, L., Luo, L., Stoleru, R., Stankovic, J. A., and Abdelzaher, T. F. (2009). Achieving long-term surveillance in vigilnet. *ACM Trans. Sen. Netw.*, 5(1):9:1–9:39.

Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. (2006). Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 381–396, Berkeley, CA, USA. USENIX Association.

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 (Proposed Standard).

Zhang, T. and Li, X. (2014). Evaluating and analyzing the performance of rpl in contiki. In *Proceedings of the First International Workshop on Mobile Sensing, Computing and Communication*, MSCC '14, pages 19–24, New York, NY, USA. ACM.

Uma solução eficiente para a leitura automática de medidores de energia usando Veículos Aéreos não Tripulados

José R. Torres Neto[1], Daniel L. Guidoni[2], Jo Ueyama[3] e Leandro A. Villas[1]

[1] Instituto de Computação – Universidade Estadual de Campinas
[2]Dept. de Ciência da Computação – Universidade Federal de São João del-Rei
[3] Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo

jrtorresneto@lrc.ic.unicamp.br, guidoni@ufsj.edu.br
joueyama@icmc.usp.br e leandro@ic.unicamp.br

Abstract. *The most electrical power suppliers to recruit meter readers to carry out collection of power consumption of the consumers. This activity is done manually and can provide error and cause financial and physical losses. Solutions have been proposed in the literature focusing on more efficient routes for meter readers or the use of a car to perform the task remotely, which can increase the cost and have delayed task by problems of transit traffic. To overcome these challenges and constraints, we propose an architecture to automatic meter reading (AMR) system using Unmanned Aerial Vehicles. Two main contributions can be absorbed of this solution: i) the meter readers are no longer needed; ii) reducing the time and distance to carry out the collection of the readings from the power meters. In addition a protocol that ensures wireless communication between the Unmanned Aerial Vehicle and the power meters was designed. The solution was validated and evaluated in an urban environment and the results achieved show a decrease up to 72.6% of time and 45.1% of the distance as compared to the use a car. The communication protocol can avoid at least 98.3% of packet collisions.*

Resumo. *A maioria das empresas fornecedoras de energia elétrica contratam leituristas para realizar a coleta dos consumos de energia dos consumidores. Tal atividade é feita manualmente e pode apresentar erros e causar prejuízos financeiros e físicos. Soluções na literatura têm sido propostas com foco em rotas mais eficientes para os leituristas ou o uso de um carro para realizar a atividade remotamente, o que pode aumentar o custo e ter a atividade atrasada por problemas no trânsito. Para superar estes desafios e limitações, este trabalho apresenta uma arquitetura de sistema automático de leitura de medidores (AMR) utilizando Veículos Aéreos não Tripulados. A solução proposta apresenta duas contribuições fundamentais: i) não é necessário a utilização de leituristas; ii) redução do tempo e da distância para a realização da coleta das leituras dos medidores de energia. Além disso, foi apresentado um protocolo para realizar a comunicação sem fio entre o Veículo Aéreo não Tripulado e os medidores de energia. A solução foi validada e avaliada em um ambiente urbano e obteve uma redução de 72.6% em relação ao tempo necessário para realizar as leituras e 45.1% da distância percorrida em comparação com a solução que utiliza o carro. Além disso, o protocolo de comunicação proposto reduz em 98.3% a colisão de pacotes durante a comunicação de dados.*

1. Introdução

Nos últimos anos, os vários avanços em tecnologias sem fio e sistemas embarcados permitiram o gerenciamento de medição inteligente para medidores das casas dos consumidores. O sistema de leitura automática de medidores (AMR - *Automatic Meter Reading*) corresponde na coleta automática de medidas de medidores de consumo de energia, água ou gás, sem a intervenção humana, com finalidade de analisar dados, faturamento ou identificação de problemas [Khalifa et al. 2011, Li Quan-Xi 2010, Brasek 2004, Siddiqui et al. 2014]. O sistema deve ter um baixo consumo de energia, baixo custo, confiável e seguro, assim pode fornecer medidas com precisão e eficiência [Boukerche et al. 2007, Cao et al. 2008b, Villas et al. 2011, Guidoni et al. 2012a, Guidoni et al. 2012b].

Importantes economias de custos logísticos e operacionais tem sido realidade em empresas distribuidoras de energia [Miao et al. 2009] com o sistema AMR, que não reduz somente os custos operacionais das empresas fornecedoras de energia, gás ou água, como também está ligado diretamente a um eficiente gerenciamento de controle de demandas e monitoramento dos dados [Park et al. 2002]. O sistema AMR tem como objetivo ser utilizado para qualquer tipo de medidor e já vem sendo utilizado em diversos países, e.g., Estados Unidos, Canadá, Austrália, Itália e outros. Consequentemente, os sistemas de leitura automática de medidores tem atraído também a atenção de pesquisadores devido a suas vantagens e benefícios [Tuna 2012].

Os sistemas AMR podem ser divididos em três categorias principais [Bittner et al. 2010, Tuna 2012]:

- *à pé*: o leiturista vai em cada casa dos clientes portando um smartphone ou dispositivo similar para a realização da coleta de dados; Esse tipo de sistema também é conhecido como *walk-by*.
- *assistido por veículo*: um funcionário dirige o veículo que possui um coletor de dados que realiza a coleta das leituras por meio de comunicação sem fio entre os medidores e o coletor de dados presente no carro; Esse tipo de sistema também é conhecido como *drive-by*.
- *utilização de redes fixas*: tais redes possuem uma infra-estrutura que conecta os medidores de energia à uma base central, sem a necessidade de intervenção humana. Dessa forma, toda a coleta é feita remotamente;

Os sistemas AMR podem ser com tecnologia com fio ou sem fio quando projetados para funcionar através de uma rede fixa [Li Quan-Xi 2010]. Porém, o sistema AMR sem fio tem um custo menor para planejamento e implantação [Cao et al. 2008b] e pode-se usar uma Rede de Sensores Sem Fio (RSSF) com o objetivo de realizar as leituras e fazer o roteamento de dados [Cao et al. 2008a, Kawai et al. 2008]. O uso de uma rede fixa pode prejudicar os clientes que estão próximos à base central, pois irão receber uma série de leituras para encaminharem, causando o aumento do seu consumo de energia com transmissões e terão um maior custo. Algumas soluções propõem o uso da tecnologia de comunicação *Power Line Carrier* (PLC), porém, ao considerar um meio barulhento, pode ocorrer uma alta atenuação do sinal e susceptibilidade à interferência de dispositivos próximos, resultando em uma alta taxa de perda, além de sua escalabilidade em sistemas AMR ser questionada por trabalhos da literatura [Khalifa et al. 2011, Siddiqui et al. 2014]. Um estudo recente realizado

por [Siddiqui et al. 2014] mostra resultados de sistemas PLC que não conseguiram operar em ambientes com ruídos de alta frequência geradas por dispositivos que utilizam técnica de comutação rápida (e.g., lâmpadas fluorescentes com reator eletrônico, fontes de alimentação chaveadas) afetando seriamente o desempenho do sistema de comunicação, além dos protocolos utilizados não serem documentados publicamente.

A abordagem mais utilizada na maioria dos países é a *walk-by*. Nessa abordagem, o leiturista deve ir em todos os medidores das casas passando por todas as ruas, o que pode demandar bastante tempo, além do tempo para realizar a leitura e o armazenamento manual dos dados dos medidores. Assim como na abordagem *walk-by*, a abordagem *drive-by* também necessita que o carro percorra todas as ruas, estando sujeito a constrangimentos, a acidentes e sinais de trânsitos. Devido a tais problemas identificados nos sistemas AMR proposto na literatura, este trabalho propõe uma leitura automática de medidores de consumo de energia assistido por um Veículo Aéreo não Tripulado (VANT). O VANT sobrevoa uma determinada área de interesse enviando mensagens de requisição solicitando as leituras de consumo de energia dos medidores. Ao receber a requisição do VANT, o medidor cria um pacote de dados com suas informações e sua leitura atual e responde à solicitação do VANT. Ao receber as informações dos medidores, o VANT armazena as leituras e pode notificar a empresa de energia, usando comunicação via satélite ou infraestrutura 3/4G, ou pode levar os dados para base com seu retorno à origem do voo.

O restante deste trabalho está estruturado da seguinte forma. Na Seção 2, é fornecido uma visão geral dos trabalhos relacionados. A solução proposta é descrita detalhadamente na Seção 3. A Seção 4 apresenta as avaliações de desempenhos e os resultados das simulações. Na Seção 5 é discutido a aplicabilidade da solução proposta. Finalmente a Seção 6 conclui o trabalho com observações e trabalhos futuros.

2. Trabalhos Relacionados

Foram realizados vários estudos propondo metodologias a fim de coletar dados de medidores automaticamente em sistemas AMR, bem como a comunicação de tais medidores [Nhan et al. 2012, Peral et al. 2012, Sharef et al. 2013, Tuna 2012]. Empresas de energia, gás e água possuem esse problema de coleta de dados, por esta razão estes estudos abrangem medidores de todos os serviços e não apenas de um único serviço. Esta Seção relata alguns trabalhos propondo metodologias de coleta de dados de consumo e está dividida em *utilização de redes fixas* e *assistido por veículo*, categorias do sistema AMR.

2.1. Trabalhos usando redes fixas

Nesta categoria existe uma rede instalada que liga todos os medidores da rede à uma central de dados a fim de realizar a coleta automaticamente dos dados de consumo dos consumidores. Um sistema AMR é proposto e avaliado por Peral et al. [Peral et al. 2012], esse sistema funciona para medidores de água baseado no padrão IEEE 802.15.4 [Han et al. 2008]. O procedimento de balizamento definidos neste protocolo é usado na transmissão e na sincronização dos dados e o sistema é alimentado por baterias. A rede do sistema utiliza uma topologia em árvore usando níveis hierárquicos para diferenciar dispositivos finais e roteadores. Todos os nós (coordenador, roteador ou

dispositivo final) que constituem a rede são equipados pelo mesmo hardware de comunicação, no entanto o nó coordenador possui uma conexão de rede IP a cabo ou sem fio usada para transmitir os dados. Os autores afirmam que a cada rede suporta até 4096 nós e todos os nós apresentam uma autonomia típica de 10 anos. Por outro lado não apresentam alternativas de transmissão de dados quando o nó coordenador perder a conexão com a rede IP, uma vez que o nó coordenador é o principal responsável pelas transmissões.

Em [Sharef et al. 2013] é proposto outro sistema AMR usado para monitorar o consumo de energia, em que é acoplado um leitor analógico emitindo os pulsos do leitor à um módulo transmissor que converte as informações para o sinal digital. Esta monitoração é feita comumente mensal até formar um ciclo, em seguida os dados são transmitidos para uma base de dados central por dispositivos com capacidade de comunicação sem fio. Os experimentos realizados neste estudos apontam que os dados foram transmitidos a uma distância de aproximadamente 91 metros, e os autores afirmam as leituras ainda foram recebidos com precisão pelo módulo receptor quando a distância estava a 200 metros, porém com um leve atraso. No entanto é necessário utilizar alguma metodologia de coleta de dados adicional a esta solução, caso a área seja extensa e exista medidores incapazes de se comunicar diretamente com a estação base.

2.2. Sistemas assistido por veículo

Nesta categoria a coleta dos dados dos consumos dos clientes é feita através de um veículo portando um dispositivo de comunicação sem fio. Um sistema AMR é proposto por [Tuna 2012] a fim de avaliar áreas ruais, onde o veículo utilizado no sistema proposto é um VANT do modelo Hexarotor, que possui asas rotativas e é similar à um helicóptero. Apesar deste modelo apresentar uma baixa autonomia de voo, a decolagem e pouso de forma vertical é uma capacidade que motivou a escolha dos autores, além do VANT poder parar no ar, que é um grande benefício em voos flexíveis. Em contextos rurais é necessário ter uma boa navegação, portanto é obrigatório ter na solução um mapa do ambiente que o VANT sobrevoa e ter também a capacidade de interpretação deste mapa. Desta forma, um sistema de localização e de navegação precisa são muito importantes no sucesso final da solução. O fornecimento de informações precisas de localização e navegação do VANT são os principais foco do artigo, de tal forma que o sistema proposto integra Sistema de Navegação por Inércia (do inglês - INS) e um Sistema de Posicionamento Global (do inglês - GPS). No entanto o artigo não considera o uso de VANTs em áreas urbanas, em que a concentração de nós é maior, e nem citam como é feita a comunicação. Um planejamento de rotas também é importante para a eficiência do VANT na conclusão da coleta de medidores.

Um sistema AMR para a coleta de medidores de energia também é proposto em Nhan et al. [Nhan et al. 2012]. Tal sistema utiliza, acoplado nos medidores, dispositivos de rede sem fio de baixo custo e com um curto alcance de 40 metros, além de sugerirem um nó móvel embutido nos transportes públicos responsável para realizar a coleta de dados dos consumos dos clientes remotamente, assim garante a redução do custo da leitura. O aproveitamento do transporte público motivou os autores, uma vez que o veículo deve se deslocar por um determinado percurso em um itinerário, de modo que não seja necessário ser empregado outro veículo exclusivo para a realização da tarefa (coleta do consumo de energia dos consumidores). Para realizar a leitura dos medidores que não estão localizados no percurso do transposte público, os autores sugerem que seja usado um

nó móvel embutido em motocicletas. Em vários cenários reais mostram que a topologia da rede pode ser desconectada, pois alguns medidores de energia podem estar localizados longe um do outros, excedendo o limite de alcance que é de 40 metros. Portanto [Nhan et al. 2012] foi utilizado na comparação da solução proposta por este trabalho, considerando que todas as ruas do cenário criado serão percorridas pelo veículo.

3. Um Sistema de Leitura Automatizada de Medidores usando VANT

A solução proposta é composta por medidores de energia comunicando-se com o VANT, onde ambos são portadores de um dispositivo com capacidade de comunicação sem fio. Uma determinada área de interesse composta por medidores de energia é sobrevoada pelo VANT que paralelamente envia requisições periódicas solicitando as informações das leituras realizadas pelos medidores. Quando um medidor recebe uma requisição do VANT, ele responde enviando o seu consumo via *unicast*, ou seja, diretamente para o VANT. O VANT envia solicitações *multicast*, pois o VANT carrega um mapa da área de interesse a ser sobrevoada e com isto é possível identificar quais medidores estão ao seu alcance para receber as suas requisições de leitura. Desta forma, também é possível saber quais medidores responderam sua requisição podendo, na próxima requisição *multicast*, enviar uma nova requisição para um nó contido em seu alcance e que não obteve resposta, uma vez que pode existir colisões de mensagens.

O sistema proposto é dividido em duas fases em que a *primeira fase* (Plano de voo) e a *segunda fase* (Protocolo de comunicação) são descritas a seguir:

1. *Plano de voo*: Percurso do VANT para sobrevoar toda a área de medidores de energia.
2. *Protocolo de comunicação*: Comunicação entre o VANT e os medidores de energia

O VANT possui um mapa geográfico que contém uma lista de pontos que o mesmo deve sobrevoar, que são os medidores de energia e suas posições. Ao sobrevoar todos os pontos listados, o VANT terá concluído seu percurso da origem ao destino. Após, o VANT retorna à base. O plano de voo é feito para uma aplicação específica de uma empresa de energia com a finalidade do VANT sobrevoar um número mínimo de pontos abrangendo toda a área de interesse. O plano de voo do VANT considera uma planta (contendo lote e ruas) em um plano cartesiano dos eixos X e Y. Desta forma, o VANT inicia seu voo na base, por exemplo, ponto(0,0), e se move até a posição do próximo ponto(X,Y) listado no mapa. Assim, o VANT se move sempre partindo de sua localização atual até percorrer todos os pontos listados.

Criar um plano de voo otimizado não é uma tarefa fácil e deve-se levar em consideração vários fatores como área de interesse, comunicação utilizada, altitude, velocidade, tempo de vida da bateria instalada no VANT e segurança. Por esse motivo, o foco principal deste trabalho é no problema de comunicação entre o VANT e os medidores de energia. Desta forma, foi utilizado o plano de voo linear nos experimentos realizados. No plano de voo linear, o VANT se move no eixo X (movimento horizontal) e no eixo Y (movimento vertical). O cenário apresentado na Figura 1 mostra a divisão de uma área por quadras (bloco de casas), em que cada quadra é composta por 16 casas, cada casa é composta por um sensor acoplado no medidor de consumo. As setas vermelhas representam a direção em que o VANT se movimenta. O VANT parte da origem (ponto(0,0)) e se move

para direita em linha reta até a extremidade horizontal (ponto (Xmax,0)). Em seguida, o VANT de desloca para baixo (eixo Y) e o valor do seu deslocamento será duas vezes o seu raio de comunicação de modo que obedeça o limite de alcance do VANT para os sensores acoplados nos medidores. Ao se deslocar no eixo Y, o VANT se move para esquerda em linha reta até a extremidade horizontal (ponto (0, Y_atual)) e depois novamente no eixo Y, completando um ciclo. O VANT repete o ciclo até chegar na extremidade da área (Xmax,Ymax) e, em seguida, retorna ao ponto de origem.

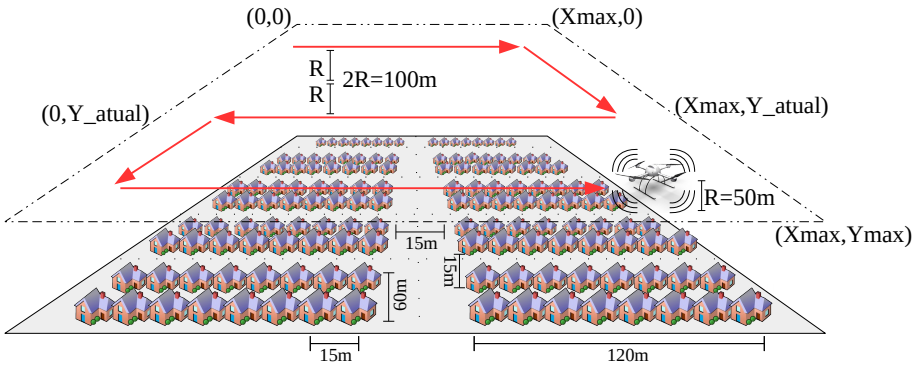


Figura 1. Plano de Voo do VANT no cenário considerado

O Protocolo de Comunicação proposto neste sistema é composto por dois tipos simples de mensagens: i) Mensagem de Requisição: são as mensagens de solicitação do VANT para os medidores, que contém uma lista ID dos medidores que devem responder; e ii) Mensagem de Resposta: são as mensagens que os medidores respondem ao VANT com sua informação de consumo e o ID do medidor.

A figura 2 mostra um fluxograma da solução proposta com os procedimentos realizados no VANT e nos medidores de energia. Os procedimentos acontecem em três ambientes: *Plano de voo, VANT* e *medidores*. As setas indicam o sentido do fluxo dos procedimentos. Ao decolar, o VANT agenda uma transmissão para solicitação de leitura dos medidores de acordo com a sua velocidade. Quando o tempo de espera para a transmissão se esgota, o *VANT* envia uma mensagem de requisição *multicast* com o objetivo de diminuir a quantidade de mensagens transmitidas. As transmissões enviadas pelo *VANT* são periódicas e paralelas ao seu *plano de voo* sobre o campo de *medidores*. Um temporizador define o intervalo de tempo t das requisições feitas pelo VANT levando em consideração a velocidade média do *VANT* v e seu raio de comunicação r, como mostra a Equação (1). A cada t o *VANT* envia mensagens de requisição para os medidores.

$$t = \frac{r}{2v} \tag{1}$$

Nos *medidores de energia*, o nó recebe a requisição do *VANT* e lê o consumo de energia atual. Em seguida, o nó responde a requisição do *VANT*. Antes dos *medidores* responderem ao *VANT*, o nó estima sua distância em relação ao *VANT* e verifica em qual *slot* está localizado e a janela de tempo do *slot*. Em seguida, o nó computa um valor aleatório dentro da janela de tempo e agenda uma transmissão com o valor obtido (Ver Subseção 3.1). Recebendo as respostas dos medidores, o *VANT* armazena as leituras de energia com as respectivas informações dos medidores.

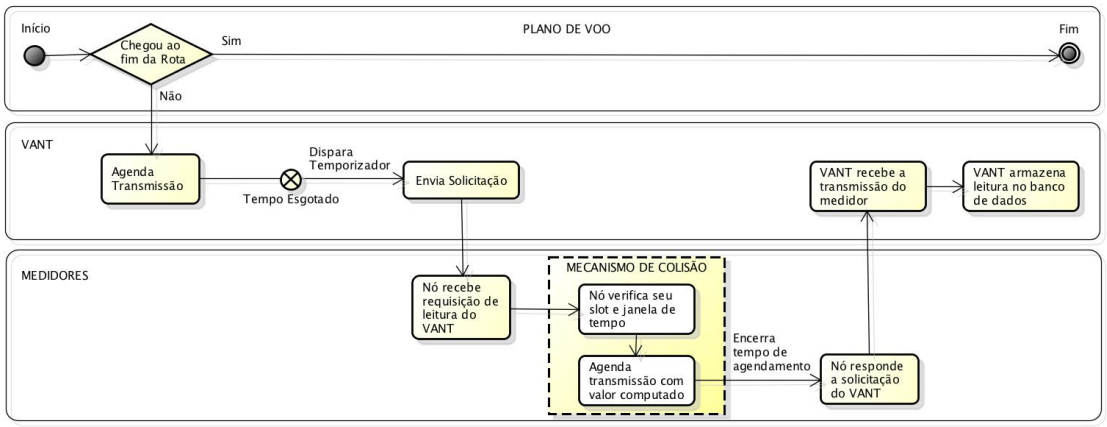


Figura 2. Principais componentes da solução proposta.

3.1. Mecanismo de Colisão

Na forma como ocorrem a troca de mensagens entre o VANT e os medidores, existem desafios que devem ser considerados, e.g. sincronizar o percurso do VANT e o tempo de resposta dos medidores de modo que evite colisões entre pacotes. Diante disso, a velocidade do VANT (nó sink) e o tempo de propagação das mensagens devem ser avaliados [Horácio A.B.F. Oliveira and Loureiro 2010]. O sistema proposto propõe um Mecanismo de Colisão a fim de evitar colisões entre pacotes. Este mecanismo pode ser detalhado da seguinte forma:

i) *Divisão do campo de alcance* : O raio de comunicação do VANT abrange uma área limitada, que dividimos em 3 *slots* de tamanho de largura igual.

ii) *Cálculo da distância estimada*: Ao receber a mensagem de requisição do VANT, cada nó calcula sua distância estimada em relação ao VANT.

ii) *Agendamento da resposta*: De acordo com a distância estimada encontrada pelo nó, é computado um valor de tempo de espera para que o nó responda a solicitação do VANT.

A *Divisão do campo de alcance* pode ser visualizado na Figura **??**, em que separa os nós por *slots* de acordo com sua distância estimada. Desta forma, cada *slot* atribui um comportamento diferente para os nós que estão contidos em sua área. Tal comportamento é decisivo no momento de resposta da requisição do VANT. É importante salientar que um nó só pode estar incluído em apenas um *slot*. Porém, em um primeiro momento, ao receber uma requisição, um nó pode estar em um determinado *slot* e em um segundo momento, ao receber outra requisição para retransmitir suas informações, o mesmo nó pode estar localizado em outro *slot* diferente do *slot* no primeiro momento. Isto acontece porque o VANT transmite requisições periódicas enquanto se desloca pela área de interesse. Além disso, isso acontece porque o VANT reenvia requisições para os nós que não obteve resposta.

Para realizar *Cálculo da distância estimada*, foi usado a técnica do indicador de intensidade do sinal recebido (RSSI) [Wu et al. 2008], que é a técnica mais utilizada para estimar a distância entre dois nós, pois utiliza apenas o rádio transmissor/receptor para a comunicação e nenhum outro *hardware*. A técnica de RSSI também não utiliza ne-

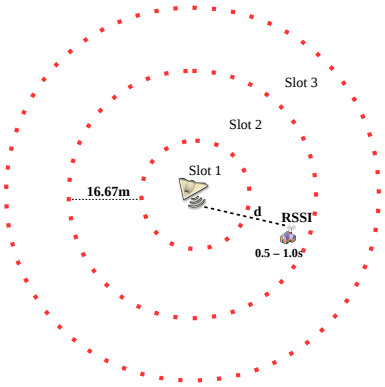


Figura 3. Mecanismos de Tratamento de colisões

nhuma mensagem de controle para estimar a distância, como é feito em outras técnicas de estimativa de distância [Villas et al. 2013, Cardoso et al. 2014]. O VANT envia sua posição atual acoplada em toda mensagem de requisição de leitura dos medidores para que o medidor estime sua distância em relação a posição do VANT usando RSSI. Esta distância estimada encontrada definirá o *slot* que o nó está localizado. A velocidade do VANT, distância estimada e tempo de propagação da mensagem são fatores importantes que precisam ser avaliados para a definição do intervalo em que o medidor responderá a solicitação do VANT. A Equação $t = \frac{d}{3108\ m/s}$ foi considerada para definir o tempo em que o VANT receberá uma mensagem dos medidores, onde t é o tempo que os medidores enviam suas leituras, d é a distância estimada do medidor em relação ao VANT e $3.108m/s$ é a velocidade de propagação de uma mensagem utilizando o protocolo IEEE 802.11.

O *Agendamento da resposta* é obtido baseado na tabela 1, onde cada *slot* possui uma janela de tempo. Uma vez obtida a distância estimada, o nó verifica em qual *slot* está localizado para efetuar sua transmissão de resposta. Conhecendo seu *slot*, o nó computa um valor aleatório dentro do intervalo estabelecido pela tabela 1. Para cada *slot*, o valor computado é o tempo em que o nó responderá a requisição do VANT. Em seguida o nó efetua a transmissão de sua leitura no valor de tempo computado.

Intervalo de Tempo	
Slot 1	1.0s - 1.5s
Slot 2	0.5s - 1.0s
Slot 3	0.0s - 0.5s

Tabela 1. Intervalo de Tempo do Mecanismo de Colisão

4. Avaliação de desempenho e Discussão dos Resultados

Os parâmetros de configuração e os resultados da avaliação de desempenho realizada por meio de simulações serão mostrados nesta Seção. É importante que o grupo já realizou experimentos reais com um VANT comunicando-se com uma RSSF no solo, como é apresentado resultados em [Faiçal et al. 2014]. Neste trabalho em particular, foram realizados experimentos com simulações utilizando o simulador Sinalgo [Sinalgo 2014]. É importante salientar que o simulador Sinalgo não simula colisão entre mensagem. Dessa forma, foi implementada uma detecção de colisão no Sinalo.

4.1. Configuração do Cenário

As simulações realizadas foram feitas com o objetivo de realizar estudos de análise do sistema de leitura automática de medidores proposto, em que o VANT sobrevoa uma determinada área de interesse coletando as informações dos medidores de consumo de energia dos clientes. Para isso, o sistema proposto utilizando um VANT foi comparado com outras duas abordagens: i) Abordagem *assistido por veículo* (Carro) - onde o veículo equipado com uma interface de comunicação sem fio passa por todas as ruas coletando as informações de consumo de energia dos clientes; e ii) Abordagem *a Pé* - onde uma pessoa caminha por todas as ruas e passa por todos os medidores da área de interesse portando um *smartphone*, ou dispositivo similar, para coletar as informações dos medidores de consumo de energia dos clientes. A leitura automática de medidores proposta por [Nhan et al. 2012] foi utilizada como base na abordagem do veículo.

No cenário criado para realizar as avaliações, foi considerado um bairro típico no Brasil com medidas 15mx30m de largura e fundo do lote e com ruas de 15m de largura (Ver Figura 1). Foi considerado também diferentes parâmetros reais nas configurações do simulador, tais como o VANT e nós sensores (MicaZ). Os parâmetros utilizados nas simulações são apresentados na Tabela 2.

Parâmetros para o VANT, Carro e a Pé	
Velocidade VANT	14 m/s
Intervalo de Requisição do VANT	1.5 s
Velocidade do Carro	7 m/s
Intervalo de Requisição do Carro	2.4 s
Velocidade da Pessoa	1.3 m/s
Raio de Comunicação (VANT, Carro, Sensor)	50 m
Número de medidores	3000, 6000, 9000, 12000 e 15000

Tabela 2. Valores dos Parâmetros de Simulação

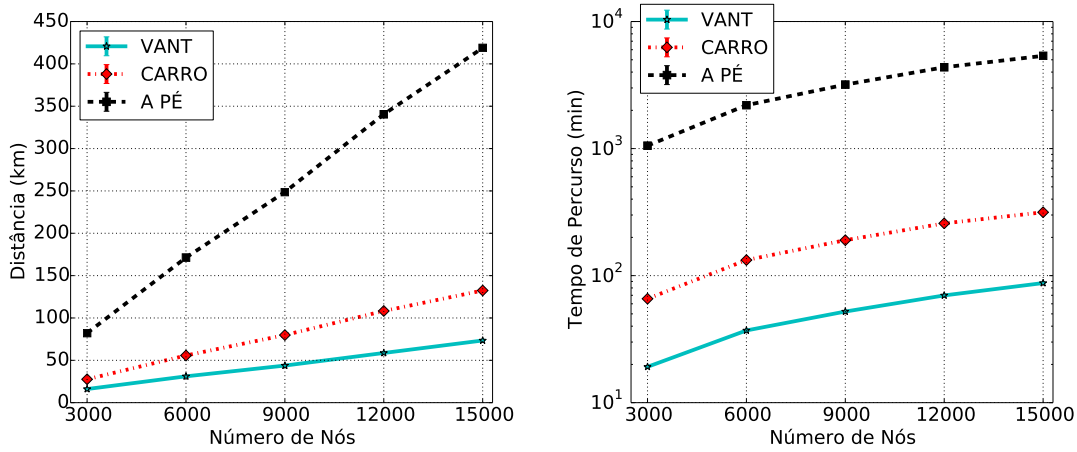
As métricas utilizadas nas avaliações de todas as abordagens foram distância e tempo (Seção 4.2). Analisamos também o desempenho da solução proposta em dois momentos com relação à transmissões efetuadas. No primeiro momento o VANT se comunica com os medidores de energia através de transmissões realizadas sem o uso do mecanismo de colisão, enquanto que no segundo momento utiliza-se o mecanismo de colisão para as transmissões. Os resultados foram obtidos considerando-se 95% de intervalo de confiança.

4.2. Resultados das simulações

A avaliação da distância percorrida por cada abordagem (*VANT*, *Carro*, *a Pé*) é apresentada na Figura 4(a), em que foi considerado diferentes tamanhos de bairros e quantidade de clientes. As rotas das abordagens *Carro* e *a Pé* têm suas rotas limitadas pelas ruas da cidade, o que não acontece no plano de voo do VANT. Isso acontece devido ao fato que a rota calculada na abordagem proposta não tem restrições topológicas, uma vez que o VANT sobrevoa a área de interesse considerando apenas os medidores dos clientes. Isto torna a distância percorrida pelo VANT menor em comparação com as outras abordagens. A Figura 4(a) mostra que quando a rede possui 3000 nós (medidores de clientes), a distância percorrida pelo VANT é 1.72 e 5.11 vezes menor quando comparado ao Carro e a Pé, respectivamente. O mesmo comportamento acontece quando comparamos a rede com

15000 nós, em que o VANT apresenta o percurso 1.80 e 5.70 vezes menor que o Carro e a Pé, respectivamente. É importante observar que ao aumentar a quantidade de nós, o tamanho da área de interesse aumenta.

A avaliação de tempo necessário para a realização da coleta dos dados dos medidores de energia feita por cada abordagem (*VANT*, *Carro*, *a Pé*) é apresentada na Figura 4(b). Os valores das velocidades utilizadas nos cenários de simulação para cada abordagem são os valores apresentados na Tabela 2. Tais valores são dados reais de um VANT disponível no mercado. O tempo para o Carro percorrer as ruas não considera congestionamento ou sinal. Com relação ao percurso do leiturista, foi considerado uma Caminhada de uma pessoa, onde o leiturista tem 10 segundos para realizar a medição e armazenar o valor. Pode ser observado na Figura 4(b) que o tempo que o VANT leva para percorrer toda a área de interesse é menor em comparação com as outras abordagens. Por exemplo, quando a rede possui 3000 nós o VANT percorre a área de interesse em um tempo 3.43 e 55.03 menor que o Carro e a Pé, respectivamente. Da mesma forma em uma rede de 15000 nós, o VANT leva 3.61 e 61.43 vezes menos tempo necessário para concluir sua rota comparando com as abordagens Carro e a Pé, respectivamente.



(a) Distância percorrida.

(b) Tempo para realizar a coleta.

Figura 4. Resultados de Simulação

A abordagem *a Pé* é a mais utilizada na maioria dos países, porém, nesta abordagem, a pessoa faz a coleta da informações dos medidores de energia ao caminhar por todas as ruas. Além do tempo da caminhada, deve-se também considerar o tempo de leitura do medidor e armazenamento das informações para cada medidor. Isto não é necessário nas abordagens do *Carro* e *VANT*, pois a coleta é realizada usando comunicação sem fio. Porém, na abordagem *Carro*, o veículo percorre todas as ruas em sua rota, o que pode existir problemas de tráfego e sinais de trânsitos, diferente do VANT que leva em consideração apenas os medidores dos clientes e não as ruas. Portanto, a abordagem do *VANT* tem a distância percorrida e o tempo de percurso menor que as outras abordagens.

4.2.1. Avaliação do sistema proposto com diferentes configurações

Nas simulações realizadas, foi considerado diferentes parâmetros de velocidades do *VANT*, e.g. $10, 12, 14, 16$ e $18 m/s$. Na Figura 5 é possível observar que quando aumenta a velocidade, o tempo para sobrevoar toda a área de interesse diminui.

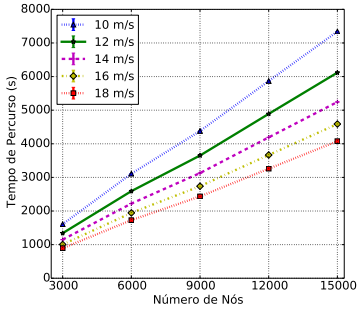
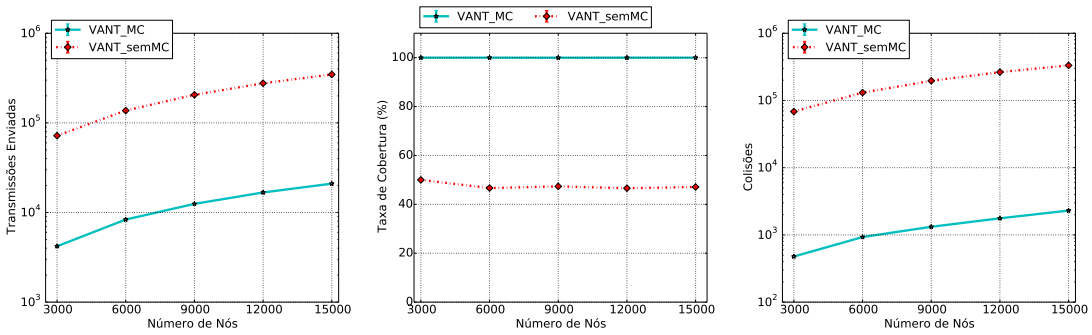


Figura 5. Velocidade do VANT

Um mecanismo de tratamento de colisões foi proposto e avaliado neste trabalho a fim de diminuir a quantidade de perdas de pacotes nas trocas de mensagens entre o VANT e os medidores de energia (ver Figura 6). A quantidade de mensagens transmitidas na solução proposta pode ser visualizada na Figura 6(a). É possível notar que utilizando o mecanismo de tratamento de colisões, a quantidade de mensagens transmitidas no meio diminui drasticamente, uma vez que o VANT transmite menos mensagens de requisição. É importante observar que considerando n o número de medidores, a quantidade de mensagens transmitidas pelo VANT é aproximadamente n quando a solução usa o mecanismo de colisão. Isto acontece porque o mecanismo proposto realmente evita colisões e o VANT precisa fazer menos retransmissões de solicitações. A Figura 6(b) mostra a taxa de cobertura obtida nos resultados com o uso do mecanismo de colisão e sem o mecanismo. Quando o mecanismo de colisão não é usado, houve uma grande perda de mensagens, e.g., cerca de 40% e 50% de cobertura de mensagens entregues. Considerando o mecanismo de controle de colisão, houve uma taxa média de cobertura a cerca de 100% de mensagens entregues. É importante observar que em ambos os casos, temos colisões. Entretanto, quando o mecanismo proposto é utilizado, existe uma maior taxa de entrega (ver Figura 6(b)) com menos colisões (ver Figura 6(c)).

5. Aplicabilidade

O modelo do VANT proposto para a solução de coleta de leitura automática de medidores de energia é o Microkopter ARF Okto XL 6S12, que está na classificação de helicópteros e é alimentado por bateria com tempo de vida de 40 minutos. Este modelo é equipado com um GPS e permite que a empresa de energia monitore o percurso do VANT em tempo real. O uso de um VANT pode melhorar também outros serviços da empresa fornecedora de energia, por exemplo, quando o sistema de energia for interrompido, a empresa pode receber essa informação pelo VANT, reduzindo o tempo da notificação de que o sistema não está funcionando. Além disso, o VANT também pode informar condições meteorológicas ou gravar vídeos de áreas com problemas.



(a) Mensagens transmitidas (b) Cobertura Atingida (%) (c) Colisão de Mensagens

Figura 6. Avaliação do Mecanismo para Evitar Colisões

Atualmente o uso do VANT em aplicações no Brasil precisa de autorização das autoridades legais, pois ainda não existe uma regulamentação concluída. O projeto de lei PL 16/15 está em andamento para a regulamentação do uso de VANTs que atendam alguns critérios como: a pesquisa e desenvolvimento científico, desde que chancelados por órgão acadêmico nacional ou apoiado pelo Ministério de Ciência e Tecnologia e Inovação. Outra preocupação com o uso de VANT são colisões com obstáculos. Entretanto, é possível acoplar um dispositivo sonar ou um *scanner* à laser a fim de identificar obstáculos. A análise do relatório dos dados coletados pode ser facilmente identificado se houve perda de dados, assim, o tempo que a empresa de energia leva para reparar o problema também pode diminuir.

6. Conclusão

Foi apresentado neste artigo uma nova solução para realização da leitura automática de medidores de energia usando Veículo Aéreo Não Tripulado e medidores de energia portando uma inteface de comunicação sem fio. Com base nos resultados apresentados, a solução proposta possui melhores resultados quando comparada com as abordagens *a Pé* e *assistida por veículo*. Os resultados mostram um ganho de pelo menos 72% e 45% em relação ao tempo gasto no percurso e a distância percorrida em comparação com as abordagens *a Pé* e *assistida por veículo*. O mecanismo de tratamento de colisão torna a troca de mensagens eficiente com ganho de 98.3%. Os resultados mostram que o sistema proposto pode ter baixo custo operacional quando comparado às soluções da literatura, além de realizar a tarefa em um menor período de tempo.

Agradecimentos

Os autores agradecem o CNPq por meio dos processos 473493/2013-6, 147356/2013-0 e 486332/2013-6, à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) e à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG).

Referências

Bittner, M., Widmer, H., Pajot, A., Alberdi, G., Hohl, H., and Kmethy, G. (2010). Energy project no 226369. Technical report, funded by the European Commission, OPEN meter.

Boukerche, A., Araujo, R., and Villas, L. (2007). A novel qos based routing protocol for wireless actor and sensor networks. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 4931–4935.

Brasek, C. (2004). Urban utilities warm up to the idea of wireless automatic meter reading. *Computing & Control Engineering Journal*, 15(6):10 – 14.

Cao, L., Tian, J., and Liu, Y. (2008a). Remote real time automatic meter reading system based on wireless sensor networks. In *Proceedings of the 2008 3rd International Conference on Innovative Computing Information and Control*, ICICIC '08, pages 591–595.

Cao, L., Tian, J., and Liu, Y. (2008b). Remote wireless automatic meter reading system based on wireless mesh networks and embedded technology. In *Embedded Computing, 2008. SEC '08. Fifth IEEE International Symposium on*, pages 192–197.

Cardoso, C., Guidoni, D., Maia, G., Ueyama, J., Loureiro, A., and Villas, L. (2014). An energy consumption aware solution for the 3d localization and synchronization problems in wsns. In *Computer Networks and Distributed Systems (SBRC), 2014 Brazilian Symposium on*, pages 376–385.

Faiçal, B. S., Costa, F. G., Pessin, G., Ueyama, J., Freitas, H., Colombo, A., Fini, P. H., Villas, L. A., Osório, F. S., Vargas, P. A., and Braun, T. (2014). The use of unmanned aerial vehicles and wireless sensor networks for spraying pesticides. *Journal of Systems Architecture - Embedded Systems Design*, 60(4):393–404.

Guidoni, D., Boukerche, A., Villas, L., Souza, F., Mini, R., and Loureiro, A. (2012a). A framework based on small world features to design hsns topologies with qos. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000732–000737.

Guidoni, D. L., Boukerche, A., Villas, L. A., de Souza, F. S., Oliveira, H. A., and Loureiro, A. A. (2012b). A small world approach for scalable and resilient position estimation algorithms for wireless sensor networks. In *Proceedings of the 10th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac '12, pages 71–78, New York, NY, USA. ACM.

Han, Y., Wang, Z., Li, L., and Zhao, Y. (2008). A fast automatic gain control scheme for ieee 802.15.4 receiver. In *Wireless, Mobile and Multimedia Networks (ICWMMN 2008), IET 2nd International Conference on*, pages 167–170.

Horácio A.B.F. Oliveira, Raimundo S. Barreto, A. L. F. E. F. N. and Loureiro, A. A. (2010). Envio de dados de consulta para sinks móveis em alta velocidade em redes de sensores sem fio. *28o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2010)*, pages 75–88.

Kawai, T., Wakamiya, N., Murata, M., Yanagihara, K., Nozaki, M., and Fukunaga, S. (2008). A sensor network protocol for automatic meter reading in an apartment building. In Miri, A., editor, *Wireless Sensor and Actor Networks II*, volume 264 of *IFIP – The International Federation for Information Processing*, pages 173–184. Springer US.

Khalifa, T., Naik, K., and Nayak, A. (2011). A survey of communication protocols for automatic meter reading applications. *Communications Surveys Tutorials, IEEE*, 13(2):168–182.

Li Quan-Xi, L. G. (2010). Design of remote automatic meter reading system based on zigbee and gprs. *Proceedings of the Third International Symposium on Computer Science and Computational Technology(ISCSCT '10)*, 2(1):186–189.

Miao, D., Xin, K., Wu, Y., Xu, W., and Chen, J. (2009). Design and implementation of a wireless automatic meter reading system. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, IWCMC '09, pages 1345–1349, New York, NY, USA. ACM.

Nhan, N.-Q., Vo, M.-T., Nguyen, T.-D., and Huynh, H.-T. (2012). Improving the performance of mobile data collecting systems for electricity meter reading using wireless sensor network. pages 241–246.

Park, B., Hyun, D., and Cho, S. (2002). Implementation of amr system using power line communication. In *Transmission and Distribution Conference and Exhibition 2002: Asia Pacific. IEEE/PES*, volume 1, pages 18–21 vol.1.

Peral, J., Merlo, E., Labrador, R., Torralba, A., Carvajal, R. G., Gil, M., Villalba, D., Grande, A., Moreno, M., and Viguera, J. (2012). Automated meter reading based on ieee 802.15.4. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 5996–6001.

Sharef, Z. T., Isa, A., Hasan, A., Toorani, A., and Yadgar, A. R. A. (2013). Automated meter reading system based on basic stamp2 microcontroller. *Asian Journal of Scientific Research*, 6(1):88–97.

Siddiqui, B., Pakonen, P., and Verho, P. (2014). Experience of communication problems in plc-based amr systems in finland. In *Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2014 IEEE PES*, pages 1–6.

Sinalgo (2014). Simulator for network algorithms. Distributed Computing Group - ETH-Zurich.

Tuna, G. (2012). Performance evaluations on uav-aided automated meter reading. *Int J Adv Robotic Sy*, 9(229).

Villas, L., Boukerche, A., Guidoni, D. L., Oliveira, H. A., Araujo, R. B., and Loureiro, A. A. (2011). Time-space correlation for real-time, accurate, and energy-aware data reporting in wireless sensor networks. In *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '11, pages 59–66, New York, NY, USA. ACM.

Villas, L., Guidoni, D., and Ueyama, J. (2013). 3d localization in wireless sensor networks using unmanned aerial vehicle. In *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*, pages 135–142.

Wu, R.-H., Lee, Y.-H., Tseng, H.-W., Jan, Y.-G., and Chuang, M.-H. (2008). Study of characteristics of rssi signal. In *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on*, pages 1–3.

Mapeando o universo da mídia usando dados gerados por usuários em redes sociais online.

Pedro H. F. Holanda[1], **Bruno Guilherme**[1], **João Paulo V. Cardoso**[2]
Ana Paula Couto da Silva[1], **Olga Goussevskaia**[1]

[1]Departamento de Ciência da Computação - UFMG
[2]Departamento de Engenharia Elétrica - UFMG

{holanda,brunoguilherme,ana.coutosilva,olga}@dcc.ufmg.br
jpcardoso@ufmg.br

Abstract. *The way people watch movies and TV has been going through great changes in recent years, one being that people are increasingly willing to share their TV watching habits with friends (and strangers) through online social network platforms. In this work, we collect usage data from the online social network tvtag and propose a data structure to represent and efficiently retrieve similarity information about movies and TV shows. We refer to this structure as "map of media", because it consists of a multi-dimensional Euclidean space, where each item is represented by a set of coordinates and the distance between them represents the similarity. We propose several metrics to evaluate the resulting structure and show that, besides computational efficiency, the proposed approach provides a high quality measure for item similarity. Moreover, the quality increases with increasing dimensionality of the map.*

Resumo. *A maneira como as pessoas consomem diferentes mídias está sofrendo grandes mudanças nos últimos anos. Uma das mais interessantes é que as pessoas desejam compartilhar seus hábitos e preferências através das redes sociais online. Neste trabalho, a partir dos dados gerados por usuários na rede social tvtag, nós propomos uma estrutura de dados para representar e consultar de forma eficiente informação de similaridade entre filmes e programas de TV. Nos referimos a esta estrutura de dados como "mapa da mídia", pois a mesma consiste de um espaço Euclidiano multidimensional, onde cada item é representado por um conjunto de coordenadas e a distância entre dois itens representa a similaridade. Nós propomos várias métricas para avaliar a estrutura resultante e mostramos que, além da eficiência computacional, a mesma provê uma medida de similaridade com alta qualidade. Além disso, a qualidade tende a aumentar com o aumento da dimensionalidade do mapa.*

1. Introdução

A maneira como as pessoas assistem a programas de TV e a filmes tem mudado drasticamente nos últimos anos [Torrez-Riley 2011]. Atualmente, as pessoas assistem aos seus programas preferidos através da internet com conexões de alta velocidade e utilizando várias fontes diferentes (Youtube e Netflix[1]). Além dos computadores, as pessoas assistem a estes conteúdos a partir de smartphones, tablets e smart TVs

[1]www.youtube.com, www.netflix.com

[Raje 2014, Sareen 2014]. Um outro ponto interessante é que a quantidade e diversidade de conteúdo disponível aumenta cada vez mais à medida que as tecnologias disponíveis para a produção de vídeo se tornam mais acessíveis.

Entre os diversos setores de produção de mídias, podemos destacar a televisão e o cinema. Assistir programas de TV é um hábito social, que gera discussões em diferentes ambientes do nosso dia-a-dia. Da mesma forma que a maneira de assistir aos programas mudou nos últimos anos, o modo de discutir e compartilhar opiniões em torno do conteúdo televisivo (ou de cinema) também sofreu modificações: as pessoas passaram a utilizar as redes sociais online como meio de propagação de opiniões e preferências [Narasimhan and Vasudevan 2012, Torrez-Riley 2011]. Para tal, são utilizadas redes sociais genéricas, como o Twitter ou redes sociais específicas, como o tvtag[2].

Apesar das mudanças nos hábitos de consumo e compartilhamento de opiniões do conteúdo da TV e do cinema, a maneira como as pessoas navegam pelas coleções existentes de programas e descobrem conteúdos novos não mudou muito. Na maioria das vezes, a navegação é realizada através de listas sequenciais, em ordens alfabéticas ou hierárquicas, ou ordenadas pela programação dos canais e guias de TV.

Poucos trabalhos na literatura propõem estratégias mais elaboradas para a navegação através de conteúdos de diferentes mídias. Podemos destacar algumas abordagens voltadas para o domínio da música [Knees et al. 2006, Neumayer et al. 2005, Goussevskaia et al. 2008, Watchmi 2014]. Os autores em [Goussevskaia et al. 2008] apresentam uma abordagem de navegação em coleções de músicas baseada no assim chamado "mapa da música", que consiste de um espaço Euclidiano multidimensional, onde cada item é representado por um conjunto de coordenadas e a distância entre dois itens representa a similaridade. Neste trabalho, estendemos esta abordagem para um novo domínio: cinema e televisão. Além disso, aprimoramos a metodologia e propomos novas métricas para avaliar a estrutura resultante no que tange o agrupamento de conteúdos similares (por exemplo, em gênero e tipo).

Para construir e analisar um sistema de organização e navegação em conteúdo baseado em similaridade, duas questões importantes devem ser respondidas: (1) Como definir similaridade entre pares de itens? Note que esta é uma questão essencialmente subjetiva, já que, o que pode parecer similar para uma pessoa, pode parecer discrepante para outra, dependendo do seu grau de expertise no assunto, por exemplo; (2) Qual estrutura de dados devemos usar para representar e consultar de forma eficiente as informações de similaridade de conteúdos, considerando robustez e adequação para navegação em grandes quantidades de conteúdo?

Neste trabalho, partimos da hipótese de que similaridade pode ser derivada a partir de dados gerados pelos usuários em uma rede social online voltada para fãs de cinema e TV, em particular o tvtag. Disponibilizada em 2010, a popularidade do tvtag teve um aumento notório: o número de usuários cresceu de 30.000 para 4.5M em 2013, com aproximadamente 500M de atividades realizadas pelos usuários[3]. Nesta rede social, os usuários podem fazer *check-ins* nos programas de TV, publicar avaliações (através de *likes* e *dislikes*) e trocar opiniões sobre um determinado conteúdo.

[2] `www.twitter.com`, `www.tvtag.com`

[3] `blog.tvtag.com`

Para a estrutura de dados de navegação, propomos o "mapa da mídia", estendendo a abordagem apresentada em [Goussevskaia et al. 2008]. Com isso, similaridade de n itens pode ser representada com complexidade espacial $O(n)$, e a similaridade entre dois itens pode ser computada com complexidade temporal constante, ou seja, $O(1)$. A partir desta estrutura, organizamos os conteúdos segundo o grau de similaridade entre os mesmos, gerando uma navegação mais aprimorada entre coleções de programas de TV e filmes, baseada em preferências dos usuários.

As contribuições deste trabalho podem ser resumidas da seguinte forma: (1) Coletamos dados sobre atividades de usuários do tvtag e fizemos uma caracterização inicial dos mesmos. Em particular, mostramos que a distribuição das atividades entre as diversas mídias segue uma lei de potência; que os usuários do tvtag tendem a ser mais positivos em relação ao conteúdo disponibilizado e tem como preferência os gêneros de comédia e animação (Seção 2); (2) Através da análise de co-ocorrência de itens em históricos de atividade de usuários, nós propusemos uma medida de similaridade para pares de itens (Seção 3); (3) A partir da medida par-a-par de similaridade, construímos uma estrutura de dados que chamamos de "mapa da mídia" (Seção 3); (4) Propusemos métricas automatizadas para avaliar a qualidade da medida de similaridade obtida e pudemos observar que a qualidade da medida aumenta com maior dimensionalidade, que dados similares são agrupados por regiões e que transições de similaridade entre itens consecutivos ao longo de trajetórias aleatórias no mapa são suaves (Seção 4); (5) Finalmente, realizamos uma análise manual dos mapas obtidos em várias dimensões, recorrendo a opiniões (subjetivas) de pessoas "comuns" e de "experts" no assunto de cinema e televisão (Seção 4). Concluímos o artigo com uma discussão de trabalhos relacionados (Seção 5) e considerações finais (Seção 6).

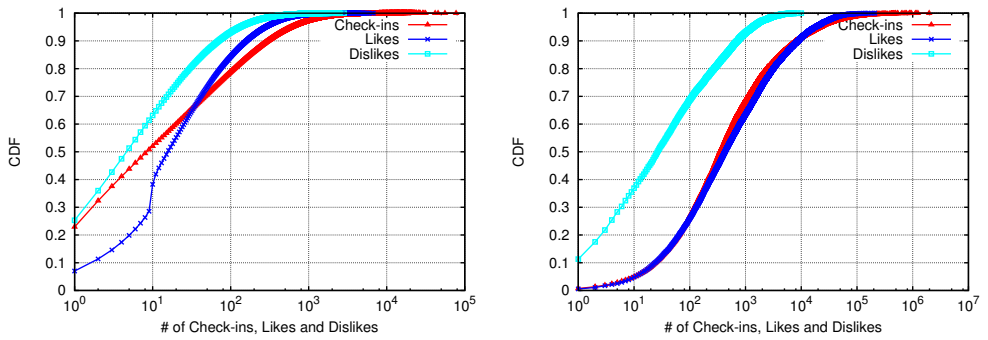
2. Coleta de Dados e Caracterização

Este trabalho é pioneiro na coleta e análise da rede social online tvtag. Assim, esta seção tem como objetivo apresentar alguns resultados importantes sobre o comportamento dos usuários e da distribuição dos gêneros dos programas de TV e filmes encontrados na nossa base de dados.

Base de dados: Para definir o mapa da mídia, duas fontes diferentes de dados foram usadas: a rede social online tvtag e o TMDB (The Open Movie Database[4]). A partir do tvtag foram coletadas informações sobre preferências e atividades dos usuários relacionadas ao conteúdo de TV e filmes, e a partir do TMBD foram obtidos metadados, tais como data de estreia, atores, diretores e gêneros dos programas de TV e filmes. A classificação de gêneros a partir do TMDB será essencial para a caracterização do conteúdo do tvtag e para a validação da estrutura de dados baseada em similaridade.

Coleta de dados: Para a coleta de dados, foram implementados dois *crawlers* Web. O primeiro *crawler* coletou as atividades dos usuários (check-ins, likes e dislikes) no tvtag. Nosso conjunto de dados representa uma boa cobertura da rede do tvtag no período entre 2011 e 2012, e consiste no total de 29M check-ins, 21M likes e 1M dislikes. O segundo *crawler* coletou os metadados disponibilizados pelo TMDB. Foram coletados 100% dos dados de programas de TV e filmes disponíveis no TMDB. Após o cruzamento dos dois conjuntos de dados, foi possível obter tanto a atividade dos usuários quanto os

[4] `www.themoviedb.org`



(a) CDF por usuário (excluindo os usuários com 0 check-ins (1M), likes (900K) e dislikes (1,6M)).

(b) CDF por programa de TV (excluindo os programas com 0 check-ins (76), likes (6) e dislikes (910)).

Figura 1. Distribuição de atividades (check-ins, likes e dislikes) no tvtag.

Usuários	1.745.000
Filmes	9.300
Programas de TV	5.000
Gêneros (TMDB)	26
Itens c/ gênero	9.977
Check-ins	92.077.000
Likes	52.776.000
Dislikes	3.033.000

Programa de TV	Likes	Check-ins	Dislikes
Big Bang Theory	206.769	1.972.968	4.935
Family Guy	206.458	459.463	6.428
Simpsons	192.276	378.779	4.449
House	180.631	405.951	3.380
Walking Dead	159.704	159.322	2.444
Glee	153.192	949.049	8.403
HIMYM	143.998	537.257	4.095
True Blood	143.888	1,238.715	5.253
South Park	139.220	153.792	5.216
Dexter	139.011	749.978	3.598

Tabela 1: (a) Dataset tvtag e TMDB (b) Top 10 programas de TV no tvtag.

metadados de aproximadamente 3.000 programas de TV e 7.000 filmes. As estatísticas relacionadas aos dados coletados estão descritas na Tabela 1(a). Por limitações de espaço, a caracterização apresentada a seguir foca nos programas de TV encontrados nos dados coletados.

Caracterização das atividades dos usuários: As Figuras 1(a) e 1(b) mostram a CDF (Função de Distribuição Acumulada) do total de check-ins, likes e dislikes realizados por cada usuário e em cada programa de TV. Para facilitar a visualização dos resultados, foram removidos dos gráficos os usuários e programas com zero check-ins, likes e dislikes. Podemos observar que quase 52% dos usuários fizeram até 10 check-ins. Aproximadamente 50% dos programas receberam até 350 check-ins. 20% dos usuários são altamente engajados na rede social, realizando entre 100 e 1.000 check-ins no período coletado e 3% dos usuários realizaram mais de 1.000 check-ins. Para termos certeza de que estes usuários não são robôs, avaliamos manualmente o comportamento dos mesmos. Segundo nossas análises, eles possuem comportamento similar aos dos usuários que interagem com outros usuários e programas de TV. Considerando o nosso conjunto de dados, uma fração considerável de programas de TV (9%) recebe mais de 10.000 check-ins. Estes valores podem ser usados como uma estimativa de audiência destes programas.

As Figuras 1(a) e 1(b) também mostram como as pessoas utilizam os

likes/dislikes. Os usuários do tvtag tendem a ser mais positivos do que negativos em relação aos programas de TV. A distribuição do número de dislikes tem um decaimento exponencial mais acentuado quando comparada as distribuições de check-in e like. Nos dados coletados, temos aproximadamente 52M likes e 3M de dislikes. Mais de 60% dos usuários atribuem mais de 10 likes. No entanto, somente 35% dos usuários atribuem o mesmo número de dislikes. A Tabela 1(b) apresenta a lista dos top 10 programas de TV considerando o total de likes.

Distribuição de Gêneros: A partir da informação coletada do TMDB, categorizamos aproximadamente 3.000 programas de TV, cadastrados no tvtag, considerando os 26 gêneros diferentes (TMDB). Nos dados coletados, 63% dos programas são classificados com um gênero; 24% com dois gêneros e 13% com três gêneros ou mais. Além disso, Drama é o gênero mais predominante no TMDB, enquanto Comédia é o favorito no tvtag. Esta diferença de porcentagem de conteúdo para diferentes gêneros pode ser explicada pelas diferentes faixas etárias da audiência. Redes sociais online tendem a atrair a audiência mais jovem quando comparado com o público da TV em geral.

3. Navegação Baseada em Similaridade

O processo para obter a estrutura de dados "mapa da mídia" a partir da informação coletada do tvtag é composto por três etapas: primeiro, estimamos similaridade item-a-item utilizando técnicas de filtragem colaborativa; segundo, construímos um grafo a partir desses valores de similaridades item-a-item; terceiro, mapeamos o grafo em um espaço Euclidiano, preservando aproximadamente as distâncias.

3.1. Similaridade Item-a-Item

Para obter valores de similaridade entre os filmes e programas de TV, nós usamos técnicas baseadas em filtragem colaborativa. O fato de dois itens serem relacionados porque co-ocorrem frequentemente em dados de uso demonstrou funcionar bem em estudos anteriores [Linden et al. 2003, Goussevskaia et al. 2008, Kuhn and Wattenhofer 2007], embora possua uma limitação em relação à amostra de usuários utilizada. Assim como a *Amazon* usa o fato de dois itens estarem relacionados por terem sido comprados pela mesma pessoa, presumimos que duas séries ou dois filmes são relacionadas caso recebam *like* do mesmo usuário, revelando suas preferências.

A simples contagem do número de ocorrência de dois itens para calcular a similaridade pareada superestima a similaridade de itens populares, uma vez que estes claramente possuem maior probabilidade de aparecerem no histórico de *likes* de um mesmo usuário devido ao seu alto número de ocorrências. Para superar esse problema, algum tipo de normalização se faz necessária. Vários coeficientes foram propostos para abordar essa questão [Matsuo et al. 2007], e o coeficiente utilizado em nossas análises foi o coeficiente cosseno, definido por $cos(i,j) = n_{i,j}/\sqrt{n_i n_j}$, onde $n_{i,j}$ é o número de co-ocorrências dos itens i e j, e n_i (n_j) o número de ocorrências individuais do item i (j).

3.2. Grafo de Similaridades

O processo descrito na seção anterior fornece um conjunto de medidas de similaridade item-a-item para aqueles itens que tiveram co-ocorrência significativa no tvtag. Note

que esse tipo de dado costuma criar matrizes esparsas, ou seja, somente uma pequena fração de todos os possíveis pares de itens terão um valor de similaridade conhecido. Para computar os valores de similaridades faltantes, nós construímos um grafo (não completo) de similaridades, onde cada vértice representa um filme ou um programa de TV, e arestas ponderadas são inseridas no grafo entre os vértices que possuem um valor conhecido de cosseno. Dessa forma, é possível computar a similaridade entre dois itens ao somar os pesos das arestas pertencentes ao caminho mínimo entre os mesmos no grafo.

O coeficiente cosseno resulta em valores próximos de zero para itens que não co-ocorrem frequentemente, e em valores próximos de um para itens que co-ocorrem muito frequentemente, por ser diretamente proporcional ao número de co-ocorrências. Entretanto, para traduzir a similaridade em uma medida de distância, o comportamento contrário seria necessário: itens mais similares ficam perto, e itens mais diferentes deveriam estar longe. Portanto, aplicamos o complemento da medida cosseno ($1 - cos(i, j)$) a todos os pares de itens coletados. Isso resulta em um grafo que contem uma aresta entre quaisquer dois itens que tenham aparecido juntos no histórico de likes de um mesmo usuário. Para evitar efeitos aleatórios, arestas com co-ocorrências abaixo de 2 foram removidas do grafo. Esse procedimento também elimina qualquer item que tenha ocorrido apenas uma vez. Em seguida, computamos a maior componente conexa do grafo, o que resultou em um (sub)grafo conexo G com 14.144 nós e $57.931.503$ arestas.

3.3. Mapa da Mídia

Calcular a similaridade entre dois itens baseando-se em um grafo de tamanho muito grande demanda um custoso cálculo de menor caminho se os vértices correspondentes não estiverem na vizinhança um do outro[5]. Esta avaliação de menor caminho não só implica em longos tempos de cálculo, mas também apresenta um consumo de memória extremamente elevado, já que o grafo deve estar presente inteiramente na memória, mesmo que se queira trabalhar com um subconjunto pequeno de itens.

Para o uso eficiente de um grafo tão grande em aplicativos (possivelmente móveis ou distribuídos), damos um passo além e criamos a estrutura de dados "mapa da mídia", que é um *mergulho*[6] do grafo em um espaço Euclidiano. Um mergulho é a atribuição de coordenadas para cada nó do grafo. Em nosso caso, a meta é preservar, aproximadamente, todas as distâncias par-a-par. Isto é, uma atribuição de coordenadas deve ser feita de forma que a razão $d_G(i, j)/d_E(i, j)$ entre a distância d_G no grafo e a distância d_E no mergulho seja aproximadamente igual a 1 para todos os pares de nós $(i, j) \in G$.

Usando a estrutura de dados "mapa da mídia", a distância (Euclidiana) entre os itens pode ser computada diretamente de suas coordenadas, ou seja, em tempo $O(1)$ e com $O(n)$ consumo de memória, onde n é o tamanho da coleção local do usuário. Nenhuma informação sobre quaisquer outros itens ou estruturas é necessária. Mergulhos são, portanto, particularmente adequados para aplicativos distribuídos e móveis. Além disso, um mergulho exibe várias vantagens funcionais, como noção de direção ou a possibilidade de medir volumes.

O processo de mergulho utiliza o procedimento de redução de dimensionalidade,

[5]Note que tipicamente itens não são vizinhos imediatos, já que o grafo de similaridades deve ser esparso. Um grafo denso, com $\Theta(n^2)$ arestas, seria potencialmente grande demais para ser armazenado.

[6]*Embedding*, em inglês

e existem diversas técnicas na literatura que realizam esta tarefa, tais como *Principal Component Analysis* (PCA) e *Multidimensional Scaling* (MDS) [Harel and Koren 2002, de Silva and Tenenbaum 2003]. Note que, o custo computacional dessas técnicas pode ser elevado e, ao se realizar o mergulho, a estrutura original dos dados pode ser perdida [Tenenbaum et al. 2000]. Para realizar o mergulho do grafo de similaridade em um espaço Euclidiano de dimensões reduzidas, nos inspiramos no algoritmo proposto em [Tenenbaum et al. 2000], denominado Isomap, que consiste em calcular a matriz de distâncias de todos os n^2 nós do grafo (conexo), usando, por exemplo, o algoritmo de Dijkstra e, em seguida, aplicar o algoritmo MDS clássico [Kruskal and Wish 1978] para reduzir a dimensionalidade do espaço Euclidiano gerado. Note que, apesar da elevada complexidade computacional desta técnica, foi possível, em algumas horas de computação (aproximadamente 6), em um servidor com 6 cores e 30GB de memória RAM, gerar o mergulho do nosso grafo G com 14.144 nós e $57.931.503$ arestas.

4. Avaliação dos Resultados

Para avaliar a qualidade da medida de similaridade representada pelo "mapa da mídia", utilizamos metadados obtidos de uma fonte independente, o TMDB. Como foi dito na Seção 2, conseguimos categorizar aproximadamente 3.000 programas de TV com 26 gêneros diferentes. Como a maioria dos itens pertencem a mais de um gênero, não seria suficiente simplesmente verificar se dois itens pertencem ao mesmo gênero. Primeiramente, definimos uma medida de similaridade de gêneros, utilizando a mesma metodologia que utilizamos para definir a similaridade entre conteúdos: dois gêneros são mais(menos) similares proporcionalmente a co-ocorrência dos mesmos em diferentes conjuntos de likes dos usuários. A similaridade entre dois gêneros g_i e g_j é dada por:

$$simGeneros(g_i, g_j) = cos(g_i, g_j) = \frac{cooc(g_i, g_j)}{\sqrt{itens(g_i)itens(g_j)}}, \quad (1)$$

onde $cooc(g_i, g_j)$ é o número de itens que foram classificados com ambos os gêneros, e $itens(g_i)$ e $itens(g_j)$ o número de itens que foram classificados pelo menos com gênero g_i e pelo menos com gênero g_j, respectivamente.

As métricas automáticas que iremos utilizar para medir a qualidade do mergulho são baseadas na premissa de que conteúdos de gêneros similares deveriam estar próximos após o mergulho no espaço Euclidiano.

Como, na nossa base de dados, a maioria dos itens é classificada com mais de um gênero simultaneamente, para definir quão similares são os gêneros de dois itens A e B, computamos a média dos cossenos entre os gêneros do item A e item B. Sejam as listas de gêneros atribuídos aos conteúdos A e B, $G_A =< g_1, g_2, \cdots g_k >$ e $G_B =< g_1, g_2, \cdots g_l >$, definimos a *similaridade por gênero* entre itens A e B como:

$$simPorGenero(A, B) = \frac{\sum_{g_i \in GA, g_j \in GB} simGeneros(g_i, g_j)}{kl}. \quad (2)$$

A seguir definimos as métricas que utilizam a definição acima.

Análise do gradiente de gêneros no espaço: Para o cálculo dessa métrica, a seguinte metodologia foi aplicada. Consideramos $p = 20$ pontos mais próximos a uma

reta, traçada entre a origem do mergulho e um ponto escolhido aleatoriamente no espaço Euclidiano (em d dimensões). A partir dos pontos escolhidos, ordenamos os mesmos pela distância ao ponto mais distante da origem da reta. Com a projeção desses pontos na reta traçada, percorremos a lista de gêneros atribuída a cada par consecutivo de pontos (conteúdos). O objetivo é verificar o quão suave é a transição entre gêneros dos conteúdos projetados na reta. Ou seja, iremos definir o gradiente de variação dos gêneros na reta, calculando a similaridade por gênero $simPorGenero(A, B)$, definida em (2), entre pares de pontos consecutivos dessa reta. Ao final, utilizamos a média dos 19 valores de $simPorGenero(A, B)$ para designar a "suavidade" da reta.

Para implementar esta métrica, realizamos o seguinte experimento: foram geradas 100 retas aleatórias, nos mergulhos de 2, 3, 5, 7 e 10 dimensões. O resultado pode ser verificado na Figura 2(a), que ilustra um gráfico do tipo "violino", que mostra uma aproximação da distribuição das similaridades por gênero entre pontos consecutivos das 100 retas aleatórias, para cada dimensionalidade do mapa. É possível verificar que o gradiente de gêneros em pares consecutivos de pontos fica cada vez mais suave com o aumento da dimensionalidade do espaço utilizado. Isso serve como indicação de que conjuntos de itens com gêneros similares são agrupados nos mapas multidimensionais, sendo o agrupamento mais coeso em espaços com maior dimensionalidade.

Predominância de um gênero em vizinhanças locais: Selecionamos aleatoriamente 50 itens em cada mergulho e verificamos a distribuição de gêneros em suas vizinhanças locais de tamanhos variados. Para cada tamanho de vizinhança, medida em número de itens contidos na mesma, computamos a similaridade por gênero média entre o ponto central e cada um dos seus vizinhos, usando a definição (2).

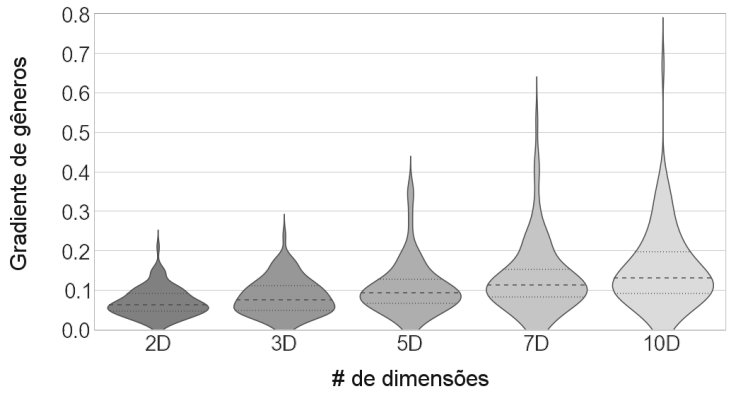
Na Figura 2(b) analisamos a similaridade por gênero entre um item e seus vizinhos no espaço Euclidiano. Mais especificamente, podemos ver a média e intervalo de confiança da distribuição de similaridade por gênero dos itens na vizinhança local, com 95% de nível de confiança. Podemos observar que em vizinhanças de tamanhos crescentes, a similaridade por gênero entre um ponto e seus vizinhos é maior em vizinhanças menores e tende a decair com o aumento da vizinhança. Além disso, observamos que este comportamento da similaridade por gênero ser maior quanto menor o tamanho da vizinhança se torna mais evidente e característico com o aumento da dimensionalidade do espaço.

Análise manual de vizinhanças locais: Escolhemos manualmente alguns filmes e programas de TV de referência, e coletamos os 9 pontos mais próximos deles nos mergulhos de 2, 3, 5, 7 e 10 dimensões. Usamos a opinião subjetiva dos autores, com a ajuda de uma especialista externa que mantém um blog sobre séries de televisão[7], para analisar qualitativamente o que acontecia com essas vizinhanças à medida que a quantidade de dimensões no mergulho era alterada.

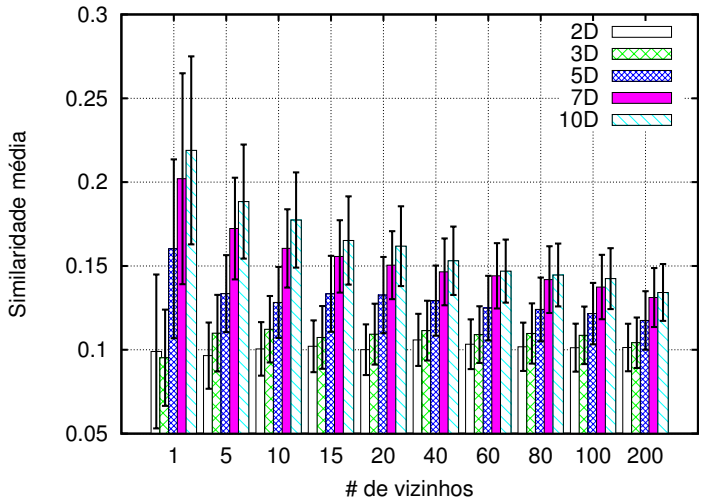
A seguir discutimos os resultados obtidos através da análise manual da vizinhança de alguns conteúdos escolhidos. Foram escolhidos 4 programas de TV e 1 filme: 2 séries de TV, 1 filme de ação e 2 *reality shows*:

(a) Série de Comédia - Parks and Recreation: A Tabela 2 mostra as vizinhanças da série para 2, 5 e 10 dimensões. Para a dimensão 2, alguns filmes de animação

[7] `www.facebook.com/tvlandbr`



(a) Gradiente de gêneros em retas aleatórias.



(b) Semelhança por gênero em vizinhanças locais.

Figura 2. Análise da qualidade dos mergulhos em função da dimensionalidade.

encontram-se na vizinhança, como Ice Age e Shrek (note que ambos pertencem ao mesmo gênero). Entretanto, ao aumentarmos as dimensões, quase toda a vizinhança é formada por séries de comédia, como It's Always Sunny in Philadelphia, 30 Rock, Weeds e Community. Esses títulos são mais representativos do público de Parks and Recreation em comparação aos dos relacionados considerando as outras dimensões, dado que as dimensões menores possuem alguns ruídos como Star Trek, um filme de ficção científica, ou Super 8, um filme de suspense produzido por Spielberg.

(b) Série Teen - Vampire Diaries: Os resultados para a análise da vizinhança desta série é mostrada na Tabela 3. Podemos observar que, para a dimensão 2, aparecem várias séries e filmes mais gerais, como Gey's Anatomy e CSI, enquanto, à medida que as dimensões aumentam, itens mais relacionados com a temática da série Vampire Diaries fazem parte da vizinhança (p.e, Twilight é um filme adolescente sobre Vampiros).

Parks and Recreation		
2D	5D	10D
Ice Age	Cosby Show	Its always sunny in philadelphia
Californication	Conan	Sons of Anarchy
Super 8	Wonder Years	Weeds
Daria	Its Always Sunny in Philadelphia	Community
Knocked up	Golden Girls	Conan
Golden Girls	Boardwalk Empire	Curb your Enthusiasm
Angel	Frasier	Anthony Bourdain no reservations
Its Always Sunny in Philadelphia	I Love Lucy	30 Rock
Shrek Third	Star Trek next Generation	Breaking Bad

Tabela 2: Vizinhança: Parks and Recreation.

Vampire Diaries		
2D	5D	10D
Hells Kitchen	Twilight Saga part 1	Twilight Saga part 1
Sex City	Smallville	Twilight
CSI Crime SceneI nvestigation	Twilight	Greys Anatomy
Greys Anatomy	CSI NY	Xfactor
White Collar	notebook	csi_miami
tangled	CSI Miami	Pretty Little Liars
Notebook	Smurfs	Pirates of Caribbean on Stranger Tides
21 Jump Street	Tangled	CSI NY
Mentalist	Greys Anatomy	Smurfs

Tabela 3: Vizinhança: Vampire Diaries.

(c) Filme de Ação - Duro de Matar: A Tabela 4 mostra a mudança da vizinhança com o aumento das dimensões do espaço Euclidiano. Podemos observar que, em duas e cinco dimensões, sua vizinhança está em uma região com alguns filmes de ficção científica, como Alien, District 9 e Back to the Future. Em 10 dimensões, os vizinhos mais próximos são Quinto Elemento, que tem inclusive o mesmo ator (Bruce Willis) como protagonista, seguido por Exterminador do Futuro, Batman e Indiana Jones, que são filmes de ação/aventura bem mais relacionados entre si.

Duro de Matar		
2D	5D	10D
American Beauty	Terminator	Fifth Element
Tron	Seven	Terminator
Terminator	Terminator 2 (judgment day)	Terminator 2 (judgment day)
Back to the Future 3	Back to the Future 2	Batman
V or Vendetta	Alien	Indiana Jones Raiders of Lost Ark
Back to the Future 2	American Psycho	Indiana Jones Last Crusade
Terminator 2 (judgment day)	Fifth Element	Back to the Future 2
Fifth Element	District 9	Indiana Jones Temple of Doom
Silence of Lambs	Batman	Alien

Tabela 4: Vizinhança: Duro de Matar.

(d) Reality Show de Competição de Moda - Project Runway: Observando a vizinhança deste programa, conforme mostrada na Tabela 5, notamos que para 2

dimensões, a vizinhança tem poucos programas relacionados: os únicos reality shows são Top Chef, que, de fato, é uma competição e compartilha o público de Project Runway, e Pawn Stars, um reality show sobre uma loja de penhores em Las Vegas. No entanto, para 10 dimensões, além do Top Chef ser o ponto mais próximo no mapa, outros reality shows relacionados aparecem, como So you think you can dance, Amazing Race e Americas' next Top Model. Este resultado mostra que o aumento das dimensões incluiu o programa num nicho que, segundo nossa especialista, atende a um público específico e bem definido.

Project Runway		
2D	5D	10D
Hey Arnold	Sweet Home Alabama	Top Chef
Brave	White Collar	Sex City
Crazy Stupid Love	Help	So you think you can dance
Top Chef	Cold case	Americas Next Top Model
Men in Black 3	Cougar Town	Late Night with Jimmy Fallon
Hawaii five0	Top Chef	Raising Hope
Pawn Stars	Private Practice	Beverly Hills 90210
Blind Side	Happy Endings	Amazing Race
Friends with Benefits	So you think you can dance	Masterchef

Tabela 5: Vizinhança: Project Runway.

(e) Reality Show de Competição de Artistas - American Idol: Analisando a vizinhança deste programa, mostrada na Tabela 6, podemos observar que, em 2 dimensões, apesar de o ponto mais próximo ser o The Voice, que é um programa muito parecido com American Idol, os outros programas não tem muito em comum além, talvez, do público alvo. Aumentando o número de dimensões, aparecem cada vez mais reality shows, como Dancing with the Stars, X Factor e America's Got Talent, e a vizinhança passa a fazer mais sentido tanto em público alvo quanto em nicho de mercado.

American Idol		
2D	5D	10D
The voice	The voice	The voice
Snow white Huntsman	Revenge	Dancing with Stars
2 broke girls	CSI NY	Ellen Degeneres Show
Twilight Saga part 1	Xfactor	Twilight Saga part 1
puss_in_boots	Twilight Saga part 1	Pretty Little Liars
Twilight	CSI Miami	Xfactor
Desperate Housewives	New Girl	Greys Anatomy
Wipeout	NCIS los Angeles	Americas got Talent
NCIS los Angeles	CSI Crime Scene Investigation	Vampire Diaries

Tabela 6: Vizinhança: American Idol.

Em suma, através da análise de vizinhanças locais dos itens apresentados é possível observar que as vizinhanças em 2 dimensões se mostraram muito genéricas. No entanto, à medida que o número de dimensões aumenta, as vizinhanças passam a fazer mais sentido, entrando cada vez mais nos nichos específicos de público que cada título atrai. Em muitos dos casos analisados em 2 dimensões, foi possível encontrar

alguma relação superficial entre títulos que poderiam ser interpretados como ruído de sobreposição (ex. gênero ou época de lançamento semelhantes).

5. Trabalhos Relacionados

TV, cinema e redes sociais online: Assistir a programas de TV e a filmes deixou de ser uma atividade restrita as casas e aos cinemas. Atualmente, as pessoas compartilham e discutem sobre estas mídias, individualmente ou em grupo, através de diferentes dispositivos e das redes sociais [Bondad-Brown et al. 2012]. Autores em [Narasimhan and Vasudevan 2012] analisam a viabilidade de usar as atividades em redes sociais (Twitter) para caracterizar o comportamento da audiência. Torrez-Riley [Torrez-Riley 2011] apresenta a evolução histórica do papel da televisão nas interações sociais. Basapur *et. al* [Basapur et al. 2012] descreve o desenvolvimento de uma ferramenta (FanFeeds) para motivar a utilização de outros dispositivos (smartphones, tablets) como extensão da TV e cinema.

Considerando o entendimento do comportamento das pessoas em relação às mídias de televisão e cinema, muitos dos trabalhos encontrados na literatura focam em redes sociais genéricas, principalmente no Twitter [Lochrie and Coulton 2012, Narasimhan and Vasudevan 2012]. Apesar do grande conjunto de trabalhos que estudam o comportamento social e as características estruturais de redes sociais online, não foram encontrados trabalhos que utilizam redes sociais direcionadas aos fãs de TV e cinema, como por exemplo o tvtag.

Medidas de Similaridade: Existem diversas estratégias para obter informação sobre o grau de similaridade entre diferentes mídias. Em relação ao conteúdo de TV e filmes, podemos citar a análise de metadados [Kim et al. 2013] e relações semânticas (baseadas em filtragem de conteúdo) [Yamada et al. 2012, Clinchant et al. 2011, Kim et al. 2013] e filtragem colaborativa [Zhang et al. 2013]. Além destas estratégias, podemos citar estratégias híbridas, que utilizam, por exemplo, filtragem colaborativa e filtragem de conteúdo [Melville and Sindhwani 2010].

Medidas de similaridades baseadas em filtragem colaborativa tipicamente exploram informação disponível ao público e, portanto, são mais escaláveis. Grande parte dos trabalhos exploram o domínio da música [Ragno et al. 2005, Goussevskaia et al. 2008]. Destacamos também [Moore et al. 2012] que apresentam uma metodologia de mergulho e predição de listas musicais baseada em *playlists* de usuários. Considerando o domínio de programas de TV, em [Zhang et al. 2013] os autores se baseiam em microblogs que podem revelar preferências dos usuários e as interações entre os mesmos. A partir da análise realizada, os autores propõem uma estratégia para sistemas de recomendação. Nosso trabalho se difere dos demais por utilizar uma rede social online direcionada aos fãs de TV e cinema para obter as informações que são utilizadas na definição da similaridade entre conteúdos.

Navegação em coleções de mídia: Em relação a navegação de coleções de mídia, muitos trabalhos estão orientados ao domínio da música [Knees et al. 2006, Neumayer et al. 2005, Goussevskaia et al. 2008]. Os artigos relacionados a programas de TV focam, principalmente, na definição da similaridade entre conteúdos. Para navegação, podemos citar a ferramenta proprietária Watchmi [Watchmi 2014] que fornece soluções para navegação e descoberta de conteúdo baseadas em similaridade de conteúdo,

utilizando metadados, preferências e redes sociais (Twitter, Facebook).

6. Discussão dos Resultados e Conclusões

Trabalhar com um mergulho ao invés de um grafo apresenta vantagens de desempenho computacional, tanto em consumo de memória quanto em tempo de processamento. Além disso, os elementos definidos em um espaço Euclidiano possibilitam várias funcionalidades interessantes, como trajetórias, volumes e a noção de direção, que podem ser exploradas na construção de novos aplicativos.

Um bom mergulho coloca os filmes, séries e programas no espaço Euclidiano de forma que títulos similares se encontrem próximos. Portanto, regiões no espaço podem ser associadas a certas propriedades dos itens, tais como gênero, época de lançamento ou público alvo, o que pode ser utilizado para diversos fins. Quando a coleção pessoal de um usuário é mapeada neste "espaço de mídia", as regiões que ela ocupa podem ser representadas como um volume (ou união de vários volumes). De maneira similar, um volume pode ser usado para definir a região de interesse de um usuário, que pode ser explorada para encontrar novos itens que esse usuário ainda não conhece.

Trajetórias, por outro lado, permitem "navegar" suavemente entre programas e regiões. O senso de direção pode ser usado para extrapolar tais trajetórias. Dado um par de filmes, podemos estabelecer uma lista de filmes que apresentem uma transição entre esses títulos. Dada uma lista de filmes, podemos definir maneiras para continuar e estender essa lista, explorando a relação entre os títulos para manter a coerência da lista.

Um mapa pode também abrir caminho para inovações dentro de casa. Ao invés de selecionar um filme navegando em menus textuais e lineares como a lista do netflix, ou percorrendo algumas estantes de DVDs em ordem alfabética em uma loja, os usuários poderão direcionar sua experiência de mídia com instruções de alto nível, como "procure um suspense engraçado", "essa série não, e nem nada parecido com ela", ou "vá em direção ao gênero *comédia francesa*" - e isso poderia retornar uma indicação tão certeira quanto uma conversa com o dono da locadora de filmes da esquina.

Agradecimentos: Ao CNPq, FAPEMIG e Lei de Informática (Projeto LG).

Referências

Basapur, S., Mandalia, H., Chaysinh, S., Lee, Y., Venkitaraman, N., and Metcalf, C. (2012). Fanfeeds: Evaluation of socially generated information feed on second screen as a tv show companion. In *Proceedings of the 10th European Conference on Interactive Tv and Video*, EuroiTV '12, pages 87–96.

Bondad-Brown, B. A., Ricea, R. E., and Pearce, K. E. (2012). Influences on tv viewing and online user-shared video use: Demographics, generations, contextual age, media use, motivations, and audience activity. *Journal of Broadcasting & Electronic Media*, 56:471–493.

Clinchant, S., Ah-Pine, J., and Csurka, G. (2011). Semantic combination of textual and visual information in multimedia retrieval. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 44:1–44:8, New York, NY, USA. ACM.

de Silva, V. and Tenenbaum, J. B. (2003). Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 705–712.

Goussevskaia, O., Kuhn, M., and Wattenhofer, R. (2008). Exploring music collections on mobile devices. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, pages 359–362.

Harel, D. and Koren, Y. (2002). Graph drawing by high-dimensional embedding. In *Revised Papers from the 10th International Symposium on Graph Drawing*, pages 207–219.

Kim, J.-M., Shin, P., Kim, J.-J., and Chung, H.-S. (2013). TV Program Retrieval System based on the Computation of Semantic Correspondence. *Journal of Next Generation Information Technology*, 4(8).

Knees, P., Schedl, M., Pohle, T., and Widmer, G. (2006). An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web. In *ACM Multimedia*, pages 17–24.

Kruskal, J. B. and Wish, M. (1978). *Multidimensional scaling*, volume 11. Sage.

Kuhn, M. and Wattenhofer, R. (2007). The theoretic center of computer science. *SIGACT News*, 38(4):54–63.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80.

Lochrie, M. and Coulton, P. (2012). Sharing the viewing experience through second screens. In *Proceedings of the 10th European Conference on Interactive Tv and Video*, EuroiTV '12, pages 199–202, New York, NY, USA. ACM.

Matsuo, Y., Mori, J., Hamasaki, M., Nishimura, T., Takeda, H., Hasida, K., and Ishizuka, M. (2007). Polyphonet: An advanced social network extraction system from the web. *Web Semant.*, 5(4):262–278.

Melville, P. and Sindhwani, V. (2010). Recommender systems. In *Encyclopedia of Machine Learning*, pages 829–838.

Moore, J. L., Chen, S., Joachims, T., and Turnbull, D. (2012). Learning to embed songs and tags for playlist prediction. In *In ISMIR*, pages 349–354.

Narasimhan, N. and Vasudevan, V. (2012). Descrambling the social TV echo chamber. *Proceedings of the 1st ACM MCSS '12*, page 33.

Neumayer, R., Dittenbach, M., and Rauber, A. (2005). PlaySOM and PocketSOMPlayer, Alternative Interfaces to Large Music Collections. In *ISMIR*, pages 618–623.

Ragno, R., Burges, C. J. C., and Herley, C. (2005). Inferring similarity between music objects with application to playlist generation. In *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*, MIR '05, pages 73–80.

Raje, N. (2014). Social tv and the "second screen". `http://trivone.com/blog/social-tv-and-the-second-screen-2/`.

Sareen, H. (2014). Why second-screen media experiences need to be social. `http://www.clickz.com`.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319.

Torrez-Riley, J. (2011). The social tv phenomenon: New technologies look to enhance televison role as an enabler of social interaction.

Watchmi (2014). Funke digital tv guide. `http://corp.watchmi.tv/about/company`.

Yamada, I., Miyazaki, M., Sumiyoshi, H., Matsui, A., Furumiya, H., and Tanaka, H. (2012). Measuring the similarity between tv programs using semantic relations. In *Proceedings of COLING 2012*, pages 2945–2960.

Zhang, Y., Chen, W., and Yin, Z. (2013). Collaborative filtering with social regularization for tv program recommendation. *Know.-Based Syst.*, 54:310–317.

Difusão de Dados em Redes de Sensores Sem Fio com Ciclo de Trabalho Reduzido *

Max do Val Machado[1,2], **Raquel A. F. Mini**[2], **Antonio A. F. Loureiro**[1]

[1]DCC/UFMG [2]DCC/PUC Minas

{maxm,raquelmini}@pucminas.br, loureiro@dcc.ufmg.br

Abstract. *Broadcasting is a challengeable task in low duty cycle Wireless Sensor Networks since when a node sends a packet, not all neighbors will be awake. Most of the solutions for broadcasting ignore that nodes can often sleep. Some work consider, but make each node share its schedule with its neighbors, spending extra energy. This work proposes and evaluates the Broadcasting for Low Duty Cycle protocol. It creates an energy-aware broadcasting tree and makes each node share information with just one neighbor that belongs to the tree. It also addresses another communication challenge that is the broadcasting when the network contains low energy regions. Simulation results show that our protocol reduces the number of transmissions, energy consumption and latency when it is compared with other existing solutions and evaluated in this work.*

Resumo. *A difusão de dados é uma tarefa desafiadora em Redes de Sensores Sem Fio com ciclo de trabalho reduzido, pois quando um nó envia um pacote, nem todos seus vizinhos estão acordados. A maioria das soluções para difusão ignoram que os nós podem ficar quase sempre dormindo. Alguns trabalhos consideraram, mas fazem com que cada nó compartilhe seu escalonamento com todos seus vizinhos, consumindo mais energia. Este trabalho propõe e avalia o protocolo Broadcasting for Low Duty Cycle. Ele cria uma árvore de difusão baseada na energia residual da rede e faz com que cada nó compartilhe informações com apenas um nó vizinho que pertence à árvore. Ele também aborda outro desafio de comunicação que é a difusão quando a rede contém regiões de baixa energia. Resultados de simulação mostram que o protocolo proposto reduz o número de transmissões, consumo de energia e latência quando comparado com outras soluções existentes e avaliadas neste trabalho.*

1. Introdução

A difusão de dados em Redes de Sensores Sem Fio (RSSFs) [Guo et al. 2013] acontece quando o nó monitor deseja enviar uma informação para todos os nós. A partir de algoritmos de difusão, o nó monitor realiza atividades tais como enviar requisições para a rede e alterar o funcionamento dos nós. Por exemplo, o nó monitor pode reduzir o ciclo de trabalho de nós localizados dentro de regiões de baixa energia, preservando a energia dos mesmos. Várias soluções para RSSFs também dependem de um mecanismo de difusão.

A difusão de dados em RSSFs é uma tarefa desafiadora quando os nós operam com ciclo de trabalho reduzido (1% ou menos), pois quando um nó envia um pacote, nem todos

*Este trabalho foi parcialmente financiado pelo CNPq e Fapemig.

os seus vizinhos estão acordados. Eventualmente, todos os vizinhos de um nó podem estar dormindo quando ele transmitir. A maioria das soluções propostas para difusão de dados nessas redes ignoram o fato que os nós podem ficar quase sempre dormindo. Por outro lado, as soluções que consideram (e.g., [Guo et al. 2013, Wang et al. 2012]) transformam cada transmissão da difusão em várias do tipo *unicasting*. Cada nó emissor agenda uma transmissão para o momento em que cada um de seus vizinhos estiver acordado. Contudo, para isso, o nó emissor tem que conhecer o escalonamento de trabalho dos seus vizinhos. Essas soluções usam uma política MAC pró-ativa [Machado et al. 2012] em que cada nó compartilha seu escalonamento com todos seus vizinhos. A principal limitação dessa política é o custo de compartilhamento gerado pelas trocas de mensagens, consumindo mais energia. Novas soluções para a difusão em RSSFs com ciclo de trabalho reduzido deveriam ser investigadas para minimizar o custo de compartilhamento.

Este trabalho propõe e avalia o protocolo *Broadcasting for Low Duty Cycle* (BroadLDC) para difusão de dados em RSSFs com ciclo de trabalho reduzido. A principal diferença entre o BroadLDC e as demais soluções conhecidas para difusão de dados em RSSFs com ciclo de trabalho reduzido é que o protocolo proposto neste trabalho faz com que cada nó sensor compartilhe seu escalonamento de trabalho com apenas um nó sensor. As demais soluções fazem com que cada nó compartilhe seu escalonamento com todos seus vizinhos. O BroadLDC cria uma árvore de difusão baseada na energia residual da rede e faz com que cada nó sensor compartilhe seu escalonamento com apenas um nó vizinho que participa da árvore disseminando pacotes. O BroadLDC também preserva a energia dos nós localizados dentro de regiões de baixa energia. O projetista de uma RSSF normalmente tem um cuidado especial com essas regiões porque o término de sua energia pode tornar a rede desconexa ou criar áreas sem sensoriamento. Resultados de simulação revelam que o BroadLDC reduz o número de transmissões, consumo de energia e latência quando comparado com outras soluções propostas na literatura e avaliadas neste trabalho. Além disso, ele pode reduzir o fluxo de dados e o consumo de energia de nós sensores localizados dentro de regiões de baixa energia para prolongar o tempo de vida da rede.

O restante deste trabalho está dividido da seguinte forma. A seção 2 aborda os trabalhos relacionados. A seção 3 mostra o protocolo proposto e a seção 4, os resultados de simulação. A seção 5 descreve as conclusões e as direções futuras deste trabalho.

2. Trabalhos Relacionados

O principal desafio da difusão em RSSFs é a entrega de pacotes porque os nós adormecem periodicamente. Alguns trabalhos (e.g., [Heissenbuttel et al. 2006]) reduzem o efeito dos nós adormecidos com o roteamento baseado no receptor no qual qualquer vizinho propaga pacotes. Contudo, essa ideia é ineficaz quando os vizinhos ficam quase sempre dormindo.

Outros trabalhos fazem com que cada nó envie um pacote diversas vezes até que um número mínimo de vizinhos o propague. No *Gossip* [Haas et al. 2006], quando um nó recebe um pacote, o nó o propaga com probabilidade p e com $1-p$, aguarda um tempo e tenta novamente. Cada nó tenta propagar o pacote até que ele ou um número mínimo de vizinhos propague. A principal limitação do *Gossip* no ciclo de trabalho reduzido é a taxa de entrega, pois os nós vizinhos quase nunca ficam acordados ao mesmo tempo.

Alguns trabalhos (e.g., [Wang et al. 2012]) transformam cada transmissão em vários *unicasting*. Guo et al. [Guo et al. 2013] propõem o *Opportunistic Flooding* (OF)

em que cada nó compartilha seu escalonamento com seus vizinhos e cada nó emissor faz suas transmissões baseado no escalonamento de seus vizinhos. A principal limitação do OF são as transmissões adicionais do compartilhamento de informações sobre vizinhos.

Outros protocolos fazem com que os nós vizinhos sincronizem seus escalonamentos para dormirem e acordem simultaneamente. [Guo et al. 2011] apresenta uma solução contendo uma árvore de difusão na qual os nós filhos seguem o escalonamento de seus pais, permitindo que cada nó emissor faça apenas uma transmissão por pacote. A limitação dessas soluções é que a sincronização se baseia em transmissões adicionais.

A literatura contém outros protocolos baseados na energia residual da rede para evitar o fluxo de dados dentro de regiões de baixa energia (e.g., [Guidoni et al. 2006]). Contudo, esses trabalhos ignoram o ciclo de trabalho reduzido.

Este trabalho compara o protocolo BroadLDC com o *Gossip* [Haas et al. 2006] e OF. Escolheu-se o *Gossip* porque ele é baseado em retransmissões como o BroadLDC e o OF, porque ele transforma cada transmissão *broadcasting* em várias *unicasting* como o BroadLDC. Este trabalho não avaliou protocolos baseados em sincronização porque as RSSFs são dinâmicas e as desatualizações limitam o uso dessas soluções. Além disso, os autores deste trabalho acreditam que qualquer sincronização na comunicação não deve considerar somente o fluxo de dados da difusão como, também, o da coleta. O trabalho também não avalia outros protocolos de difusão que usam a energia residual da rede porque os protocolos que usam essa informação ignoram o ciclo de trabalho reduzido.

3. *Broadcasting for Low Duty Cycle*

Esta seção propõe o protocolo *Broadcasting for Low Duty Cycle* (BroadLDC) e está organizada como descrito a seguir. A seção 3.1 descreve o funcionamento básico do BroadLDC. Esse funcionamento pode ser modelado a partir de eventos e ações conforme descrito na seção 3.2. A seção 3.3 apresenta como o protocolo proposto identifica regiões de baixa energia. Cada nó sensor decide probabilisticamente como será sua forma de participação na árvore e a seção 3.4 detalha o cálculo dessa probabilidade de decisão.

3.1. Funcionamento Básico

O funcionamento básico do BroadLDC é baseado em uma árvore de difusão. Ele constrói essa árvore de forma probabilística a partir do mapa de energia da rede e faz com que cada nó sensor compartilhe seu escalonamento de trabalho com somente um nó vizinho que participa da árvore como nó interno. Cada nó interno armazena o escalonamento de trabalho de todos seus filhos e é responsável por enviar os pacotes de difusão quando cada um deles estiver acordado. Os nós folhas não armazenam informações sobre vizinhos. O nó monitor é a raiz da árvore e ele pode criá-la, recriá-la quando julgar necessário e utilizá-la em outras difusões. Outras aplicações também podem se beneficiar da árvore ou das informações sobre escalonamento de vizinhos compartilhadas.

O nó monitor cria a árvore enviando um pacote de criação para seus vizinhos. Quando um nó recebe esse pacote, ele decide se será um nó interno. Se verdadeiro, ele também envia o pacote de criação para seus vizinhos, aumentando o tamanho da árvore. Cada nó interno deve descobrir o escalonamento de trabalho dos seus vizinhos e agendar um momento em que esses acordarão simultaneamente para decidir quem também será nó interno. Nesse caso, cada vizinho acorda oportunisticamente no momento combinado,

aguarda pela decisão conjunta e, depois, retoma normalmente seu escalonamento de trabalho na posição do ciclo operacional em que ele estaria se não tivesse participado da decisão. Essas descobertas e agendamentos são baseados na retransmissão de um pacote para a criação da árvore. Como mostrado em [Machado et al. 2012], o projetista da rede define o número de retransmissões e o intervalo entre elas a partir do ciclo de trabalho dos nós. O BroadLDC constrói sua árvore com nós distantes entre si com o objetivo de reduzir o número de nós internos na árvore e, consequentemente, as descobertas e agendamentos.

3.2. Eventos e Ações

Esta seção modela o funcionamento do BroadLDC baseado em eventos e ações de tal forma que quando um evento acontece, ocorre uma ou mais ações correspondentes. Os eventos considerados no BroadLDC são "receber pacote", "filiação" e "decisão". As figuras 1 à 7 ilustram as ações correspondentes do BroadLDC para cada um desses eventos.

O evento "receber pacote" acontece quando um nó recebe um pacote. O primeiro passo da ação de receber pacote é verificar se o nó corrente é o próximo nó do pacote recebido (figura 1, ponto *A*). Se verdadeiro, o nó analisa o tipo do pacote (figura 1, pontos *B*, *E* e *G*) e executa uma ação específica para receber cada um dos tipos de pacote (figura 1, pontos *C*, *F*, *H* e *I*). Em seguida, o nó termina a ação (figura 1, ponto *D*). Um pacote pode ser dos tipos dados, criação da árvore, filiação e reconhecimento. O primeiro contém dados de difusão. O nó monitor usa o segundo tipo para criar a árvore. O pacote de filiação permite que cada nó filho se apresente para seu respectivo pai que reconhecerá seu novo filho enviando um pacote de reconhecimento como resposta. Caso o nó corrente não seja o próximo nó do pacote recebido (figura 1, ponto *A*), esse nó apenas descarta o pacote (figura 1, ponto *J*). Observa-se que todos os nós da rede serão o próximo nó dos pacotes de dados, pois o BroadLDC é um protocolo para a difusão de dados.

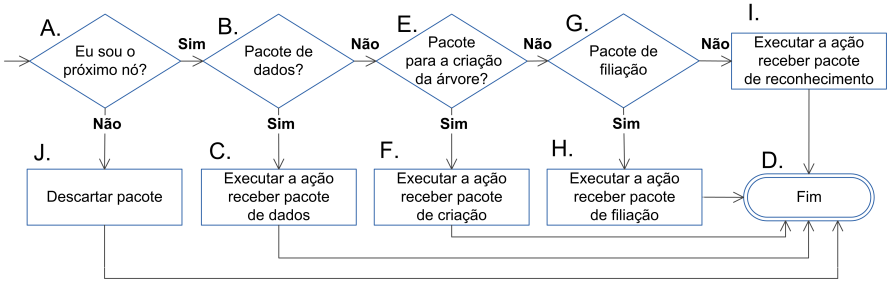


Figura 1. Ação receber pacote

A ação para receber um pacote de dados verifica se o pacote está sendo recebido pela primeira vez (figura 2, ponto *A*). Se verdadeiro, o BroadLDC encaminha o pacote para a camada de aplicação (figura 2, ponto *B*). Em seguida, se o nó corrente é um nó interno (figura 2, ponto *C*), ele escalona o envio do pacote recebido para o momento em que cada um dos seus filhos estiver acordado (figura 2, ponto *D*) e termina ação (figura 2, ponto *E*). A ação para receber pacote de dados também termina quando o nó não é interno (figura 2, pontos *C* e *E*). Se o nó recebeu anteriormente o pacote (figura 2, ponto *A*), ele descarta o pacote recebido e termina a ação (figura 2, pontos *F* e *E*).

A ação para um receber pacote de criação também verifica se o pacote está sendo recebido pela primeira vez (figura 3, ponto *A*). Se verdadeiro, o nó corrente assume que o nó anterior do pacote é seu pai (figura 3, ponto *B*). Em seguida, ele executa a ação

"filiação" (figura 3, ponto *C*) na qual o nó envia um pacote de filiação para seu pai saber sobre o novo filho. O passo seguinte faz com que o nó corrente verifique se ele está localizado dentro de uma região de baixa energia (figura 3, ponto *D*). A seção 3.3 descreve como um nó faz essa verificação. Se o nó estiver localizado dentro de uma dessas regiões, ele adapta seu comportamento para economizar energia e prolongar o tempo de vida da rede (figura 3, ponto *E*). Por exemplo, o nó pode reduzir seu ciclo de trabalho para dormir por mais tempo ou decidir que não será um nó interno para eliminar descobertas e agendamentos. Em seguida, o nó corrente verifica se sua distância em relação ao nó anterior do pacote é maior que um limiar de distância (figura 3, ponto *F*). Se verdadeiro, ele escalona o evento "decisão" para decidir se será um nó interno (figura 3, ponto *G*). Caso contrário, ele será uma folha (figura 3, ponto *I*) porque está localizado próximo ao anterior e suas descobertas seriam redundantes com as desse nó. O fato dos nós "próximos" ao anterior se tornarem folhas aumenta a distância média entre os nós internos, reduz o número desses nós na árvore e minimiza as descobertas e os agendamentos. Somente os nós internos realizam essas tarefas. Quando um nó decide ser folha, ele cancela eventuais escalonamentos do evento "decisão" (figura 3, ponto *J*) e termina a ação corrente (figura 3, ponto *H*). O nó deve cancelar esse evento porque sua decisão está tomada e ele pode ter escalonado tal evento quando recebeu o pacote de criação pela primeira vez.

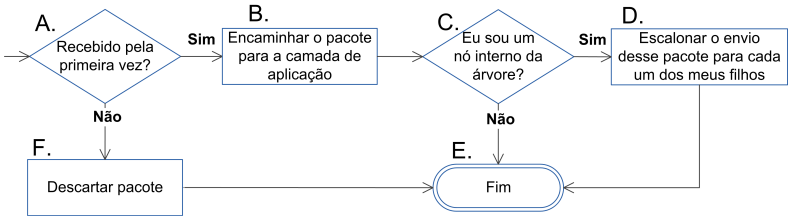


Figura 2. Ação receber pacote de dados

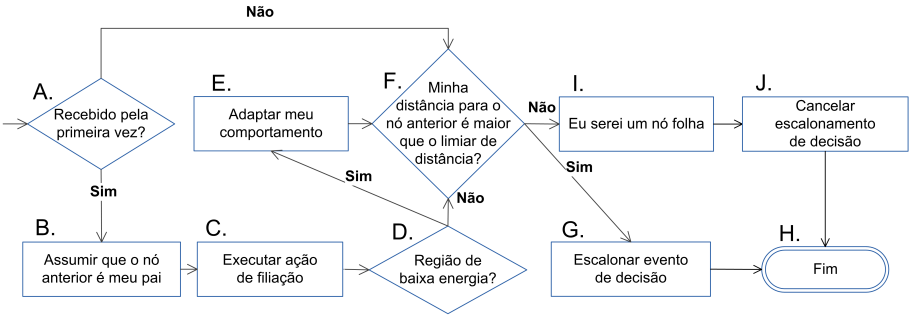


Figura 3. Ação receber pacote de criação

As ações para receber os pacotes de filiação e reconhecimento e a correspondente ao evento "filiação" fazem com que cada nó tenha um pai. Todo pai é um nó interno e responsável por enviar os pacotes de difusão quando cada um de seus filhos estiver acordado. As figuras 4 à 6 ilustram respectivamente essas ações. A ação para receber uma pacote de filiação acontece porque o nó corrente enviou um pacote de criação e recebeu como resposta outro de filiação, indicando que algum vizinho assumiu o nó corrente como pai. Essa ação faz com que o nó corrente atualize sua lista de filhos e escalone o envio de um pacote de reconhecimento destinado ao novo filho (figura 4). A ação para receber um pacote de reconhecimento faz com que o nó saiba que foi reconhecido por seu pai. Assim, ele marca esse reconhecimento e cancela o escalonamento do evento "filiação" (figura 5). Um nó executa a ação "filiação" quando recebe um pacote de criação pela primeira vez.

Nessa ação, o nó escalona o envio de um pacote de filiação para que o pai saiba sobre esse novo filho (figura 6, ponto *A*). Esse evento acontece no momento em que o pai estiver acordado. Isso é possível porque o cabeçalho do pacote de criação contém o escalonamento de trabalho do pai. O pacote de filiação também tem esse campo para que o pai saiba quando enviar pacotes para seu novo filho. Como mostrado em [Machado et al. 2012], as informações sobre escalonamento de trabalho podem ser descritas em alguns bytes. Em seguida, o nó corrente reescalona o evento (figura 6, ponto *B*). Isso porque se o nó não receber seu reconhecimento, ele precisa enviar o pacote de filiação novamente.

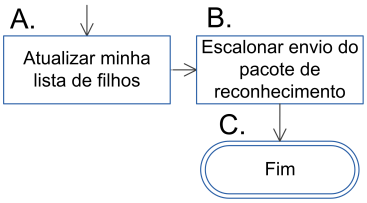


Figura 4. Ação receber pacote de filiação

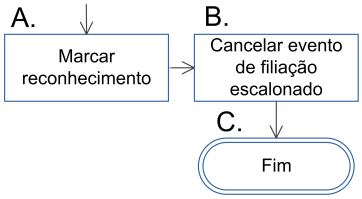


Figura 5. Ação receber pacote reconhecimento

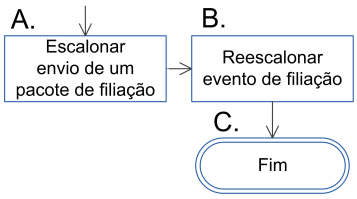


Figura 6. Ação de filiação

O evento "decisão" acontece simultaneamente em todos os vizinhos de um nó interno. Essa simultaneidade ocorre porque quando um nó envia um pacote de criação, ele específica no cabeçalho em quanto tempo seus vizinhos devem escalonar o evento "decisão". Quando esse evento acontece, o nó acorda e permanece assim até tomar sua decisão. O primeiro passo da ação de decisão faz com que o nó corrente verifique se ele participa da árvore (figura 7, ponto *A*). Se verdadeiro, ele calcula sua probabilidade de decisão, escolhe um número aleatório e verifica se esse número é menor que sua probabilidade de decisão (figura 7, pontos *B*, *C* e *D*). A seção 3.4 mostra como cada nó calcula essa probabilidade. Se o número aleatório for menor que a probabilidade de decisão (figura 7, ponto *D*), o nó será um nó interno (figura 7, ponto *E*). Assim, ele escalona suas transmissões do pacote de criação e termina a ação corrente (figura 7, pontos *F* e *G*). Essas transmissões possibilitam que o novo nó interno faça suas descobertas e agendamentos. Quando o número aleatório é maior ou igual à probabilidade de decisão (figura 7, ponto *D*), o nó reescalona o evento de decisão para acontecer em um *time step* (figura 7, ponto *H*) – tempo necessário para que um nó envie um pacote de criação e todos seus vizinhos que estiverem acordados recebam e tratem o pacote. Durante o tempo de espera, se um nó receber um pacote de criação e a distância entre esse nó e o emissor do pacote for menor que o limiar de distância, a ação "receber pacote de criação" faz com que o nó seja uma folha, terminando a decisão. Após o tempo de espera, se o nó não tomou sua decisão, ele executa novamente a ação corrente, ganhando uma nova oportunidade de decisão. Um nó reescalona o evento de decisão até que ele ou um vizinho "próximo" decida ser um nó interno. O projetista da rede pode definir um número máximo de reescalonamentos, eliminando uma repetição infinita. Quando o evento de decisão acontece em um nó interno, esse envia um pacote de criação (figura 7, pontos *A* e *I*), fazendo com que vizinhos próximos sejam folhas o que minimiza o número de nós internos na árvore.

Como mostrado em [Machado et al. 2012], para que todos os vizinhos de um nó interno tenham a oportunidade de receber o pacote de criação com o menor número possível de retransmissões, cada nó interno escalona suas retransmissões baseadas em dois pressupostos. Primeiro, o tempo total do processo de retransmissões deve ser igual

ao período em que um nó permanece adormecido em seu ciclo operacional, garantindo que todos os vizinhos do nó interno sempre acordam durante o processo. Segundo, o intervalo entre as retransmissões deve ser igual ao período em que os nós permanecem acordados em um ciclo operacional, garantindo que quando um vizinho acorda, o nó interno sempre efetua pelo menos uma retransmissão antes desse vizinho voltar a dormir.

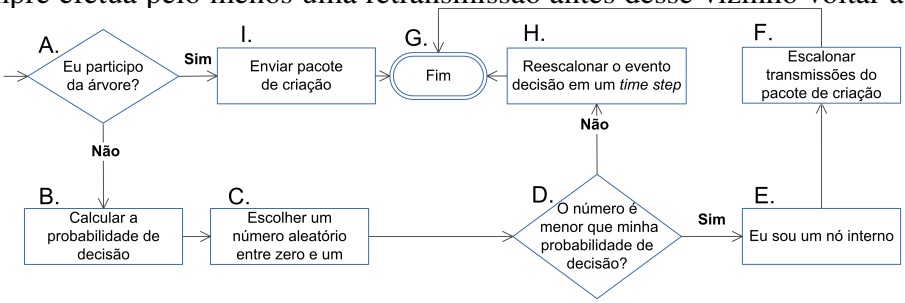


Figura 7. Ação de decisão

3.3. Regiões de Baixa Energia

Um dos principais desafios em RSSFs são as regiões de baixa energia porque o término da energia dos nós localizados dentro das mesmas pode tornar a rede desconexa ou criar áreas sem sensoriamento. O BroadLDC identifica tais regiões através de um parâmetro chamado de energia de corte. O nó monitor calcula esse valor a partir do mapa de energia da rede (informação sobre a energia residual nas diversas partes da rede) e o insere no cabeçalho dos pacotes de criação. Um nó sensor está localizado dentro de uma dessas regiões se sua energia for menor que a de corte. O restante desta seção mostra como o BroadLDC constrói o mapa de energia e calcula a energia de corte.

O mapa de energia pode ser útil para preservar a energia das RSSFs. Pode-se construí-lo a partir de uma abordagem ingênua em que cada nó envia periodicamente o valor de sua energia para o nó monitor. Contudo, o custo de comunicação dessa abordagem provavelmente a torna inviável para RSSFs. Mini et al. [Mini et al. 2005] apresentam uma solução baseada na predição do consumo de energia dos nós sensores. Cada nó sensor usa um mecanismo baseado em Cadeia de Markov para prever de forma eficiente a quantidade de energia que ele consumirá no futuro. Em seguida, o nó envia para o nó monitor os parâmetros descrevendo seu consumo de energia. Com essa informação, o nó monitor pode atualizar localmente sua informação sobre a energia atual de cada nó sensor. Este trabalho usa a abordagem de [Mini et al. 2005] para construir o mapa de energia.

Antes de enviar um pacote de criação, o nó monitor calcula a energia de corte da rede e a insere no cabeçalho do pacote. A energia de corte é um limiar usado por cada nó sensor para determinar se ele está localizado dentro de uma região de baixa energia. Este trabalho considera que uma região é de baixa energia quando a energia média de todos os nós que a cobrem é menor que a energia média da rede ($\overline{e}$) menos o desvio padrão (s).

O nó monitor utiliza o mapa de energia para calcular $\overline{e}$ e s. Em seguida, ele calcula um coeficiente de energia e_i para cada nó i que corresponde à soma da energia do nó i com a de todos os n vizinhos de i dividido por $n + 1$. Dado $\overline{e}$, s e cada e_i, a energia de corte será igual ao maior coeficiente que seja menor que $\overline{e} - s$. Quando esse coeficiente não existe, a rede não tem regiões de baixa energia e a energia de corte é zero.

3.4. Probabilidade de Decisão

O BroadLDC calcula a probabilidade de decisão de forma que os nós mais distantes do emissor do pacote e com as maiores reservas de energia tenham mais chances de pertencer à árvore. Aumentando a distância entre os nós, reduz-se o número de nós internos, descobertas e agendamentos. Além disso, o fato dos nós sensores com as maiores reservas de energia terem mais chances de serem nós internos balanceando a energia residual da rede.

A equação (1) mostra como calcular a probabilidade de decisão. Sendo, *distância*, a distância entre o nó corrente e o anterior; *raio*, a maior distância que uma transmissão pode entregar um pacote; *energia*, a energia corrente do nó; *energia max*, o maior valor de energia que um nó operando com o ciclo de trabalho padrão pode ter dado o tempo de vida corrente da rede e; $p_{\max}$, o maior valor possível da probabilidade de decisão.

$$\textit{probabilidade de decisão} = \frac{\textit{distância}}{\textit{raio}} \times \frac{\textit{energia}}{\textit{energia max}} \times p_{\max} \quad (1)$$

4. Resultados de Simulação

Esta seção avalia o comportamento do BroadLDC em cenários de difusão de dados em RSSFs com ciclo de trabalho reduzido e o compara com outras soluções para difusão propostas na literatura. Considerou-se cenários com e sem regiões de baixa energia. A seção 4.1 mostra os parâmetros de simulação. A seção 4.2 discute os resultados de simulação obtidos pelos protocolos avaliados quando a rede não contém regiões de baixa energia. A seção 4.3 possui regiões de baixa energia geradas no início da simulação.

4.1. Parâmetros de Simulação

Todos os protocolos foram implementados no *Network Simulator 2* e os resultados de simulação correspondem à média aritmética de n simulações, onde n é o menor tamanho de amostra que proporciona o intervalo de confiança desejado. O nível de confiança foi de 95% e utilizou-se o *teste-T* [Jain 1991] com $0,05$ de significância para afirmar que um protocolo é melhor ou pior que outro para uma dada métrica.

Em todas as simulações, utilizou-se uma rede conexa com nós estáticos, homogêneos e com capacidade de energia finita e não renovável. O consumo de energia de cada nó é baseado no Mica2 [Mic 2013], sua energia inicial foi suficiente para que nenhum nó morresse por falta de energia durante a simulação e seu raio de comunicação foi de [100]m. Os nós operaram com um ciclo de trabalho reduzido de 1% e foram depositados de forma aleatória em uma área [1000×1000]m^2. Considerou-se que cada nó conhece sua localização [Arshad et al. 2012]. O número de nós no campo de sensoriamento foi um parâmetro alterado em cada simulação. Esta seção avaliou a rede com 250, 500, 750 ou 1000 nós. Como a área de sensoriamento é constante, o aumento do número de nós implica no aumento da densidade da rede (número médio de vizinhos) e do número de destinos para cada difusão.

A rede contém um nó monitor sem restrição de recursos, localizado no canto inferior esquerdo da rede e cujo ciclo de trabalho é 100%. Considera-se que ele conhece as coordenadas de todos os nós sensores. Durante cada simulação, ele envia quinze mensagens de difusão distribuídas uniformemente ao longo do tempo de simulação. O tempo total de simulação é de [1000]s. Contudo, a primeira difusão acontece somente após

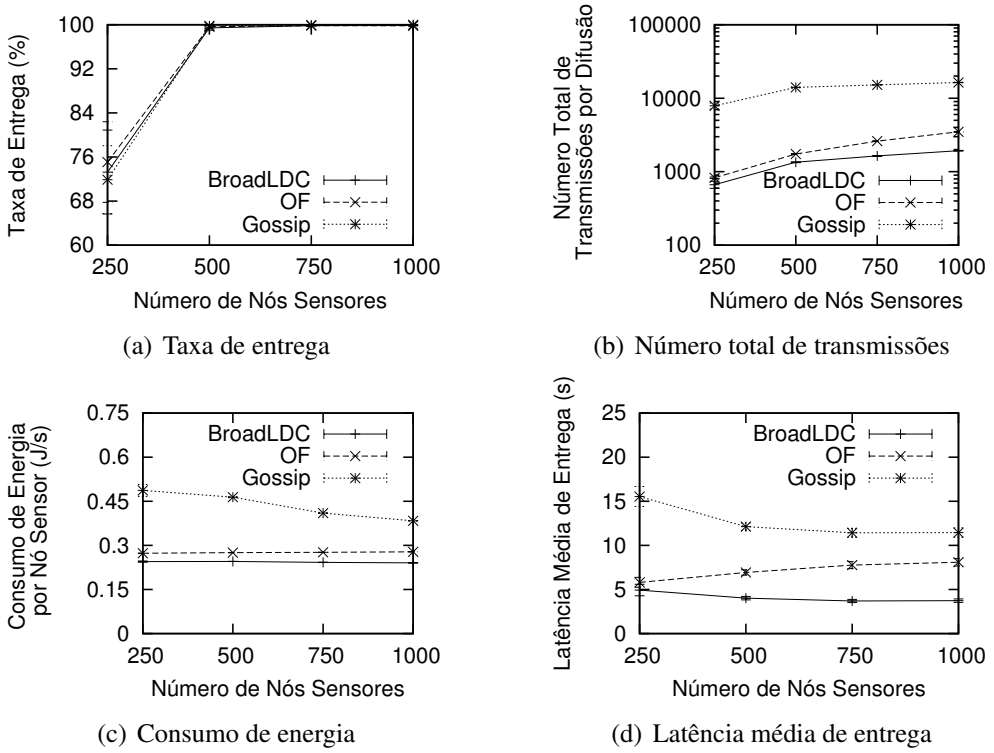
[50]s, permitindo a inicialização dos protocolos avaliados. A última acontece antes de [950]s para que a rede faça o roteamento de todos os pacotes.

As regiões de baixa energia são círculos cujos centros são escolhidos aleatoriamente e o comprimento do raio depende do número de regiões no campo do sensoriamento. Em todas as simulações, a soma das áreas dessas regiões é constante e igual a 10% do campo do sensoriamento. Aumentando-se o número de regiões, reduz-se o raio e a área das mesmas. A criação dessas regiões acontece no início da simulação fazendo com que a energia dos nós localizados dentro delas seja metade do valor inicial dos demais nós.

Esta seção compara o BroadLDC com o *Opportunistic Flooding* (OF) e o *Gossip*. Avaliou-se o OF porque ele específico para a difusão de dados em RSSFs com ciclo de trabalho reduzido. Como o OF usa uma política MAC pró-ativa para compartilhar o escalonamento dos nós vizinhos, este trabalho seguiu a proposta de Guo et al. [Guo et al. 2013] para minimizar o custo de compartilhamento na qual cada nó envia seu escalonamento somente para uma fração de vizinhos. Entretanto, como essa fração depende da densidade de nós na rede, para as simulações com 250, 500, 750 e 1000 nós, a fração de compartilhamento do OF foi 1, 0.5, 0.4 e 0.3, respectivamente. Escolheu-se empiricamente esses valores porque eles são os menores que não reduzem a taxa de entrega do OF. A fração de compartilhamento do OF reduz a taxa de entrega se os nós descartarem pacotes porque não conhecem o escalonamento dos seus vizinhos. O compartilhamento do OF acontece apenas no início de cada simulação. Avaliou-se o *Gossip* devido à sua eficácia, simplicidade e robustez para difusão de dados. Contudo, como ele apresenta uma taxa de entrega reduzida quando os nós operam com ciclo de trabalho reduzido, inseriu-se os agendamentos do BroadLDC no *Gossip*. Quando um vizinho acorda no momento agendado, ele propaga o pacote com probabilidade p e, com $1 - p$, ele aguarda um *time step* e tenta novamente. O nó repete esse processo até que ele ou outro vizinho propague o pacote. A probabilidade de decisão do *Gossip* foi calculada como no BroadLDC. Em ambos os protocolos, o limiar de distância foi 70% do raio de comunicação, o valor máximo da probabilidade de decisão foi $0,001$ e a energia máxima foi calculada a partir do comportamento do Mica2 [Mic 2013]. A adaptação do comportamento dos nós localizados dentro das regiões de baixa energia pelo BroadLDC consistiu em reduzir o ciclo de trabalho desses nós pela metade e proibi-los de ser nós internos, eliminando descobertas e agendamentos dentro das regiões. Considerou-se o custo integral da construção do mapa de energia no BroadLDC. O BroadLDC cria a árvore de difusão no início de cada simulação.

4.2. Difusão de Dados sem Regiões de Baixa Energia

Esta seção avalia o BroadLDC em um cenário de difusão no qual os nós operam com um ciclo de trabalho reduzido e a rede não contém regiões de baixa energia. A figura 8 mostra o desempenho do BroadLDC quando se aumenta o número de nós na rede. As métricas avaliadas são a taxa média de entrega, número total de pacotes transmitidos, consumo de energia e latência. A taxa média de entrega significa a porcentagem de pacotes entregues aos seus destinos e ela é fundamental porque os pacotes podem ser perdidos durante a difusão. O número total de pacotes transmitidos e o consumo de energia indicam o custo dos protocolos. A latência é um parâmetro crucial porque o atraso pode causar congestionamentos e perdas, principalmente quando os nós operam com um ciclo de trabalho reduzido no qual os emissores esperaram mais tempo até que o próximo nó acorde.



(a) Taxa de entrega

(b) Número total de transmissões

(c) Consumo de energia

(d) Latência média de entrega

Figura 8. Métricas avaliadas no cenário sem regiões de baixa energia

A figura 8-a mostra a taxa de entrega do BroadLDC, OF e *Gossip* à medida que se aumenta o número de nós no campo sensoriamento. Quando a rede tem 250 nós, os protocolos avaliados apresentam uma taxa de entrega reduzida apesar da topologia ser conexa. Esse resultado acontece porque nos três protocolos, para evitar inundação, nem todos os nós propagam um pacote recebido pela primeira vez. Assim, em redes esparsas, como alguns nós possuem apenas um caminho até o nó monitor, se um nó desse caminho não propagar uma mensagem, os demais nós do caminho não a recebem. A partir de 500 nós, os três protocolos entregam os pacotes para quase todos os nós da rede. Aplicando o *teste-T* com $0,05$ de significância, não existe diferença entre os resultados obtidos pelos protocolos avaliados à medida que o número de nós sensores aumenta.

A figura 8-b mostra o número total de transmissões com escala logarítmica no eixo y. Na figura, o aumento do número de nós no campo de sensoriamento faz com que os três protocolos avaliados façam mais transmissões. Isso acontece porque cada difusão tem mais destinos. O *Gossip* faz em média 6,17 e 9,58 vezes mais transmissões que o OF e o BroadLDC, respectivamente, porque no *Gossip*, em cada difusão, todos os nós emissores fazem agendamentos. Quando a rede tem 250 nós, o OF faz 1,24 vezes mais transmissões que o BroadLDC e quando ela tem 1000 nós, essa diferença aumenta para 1,81 vezes. O aumento dessa diferença acontece porque no OF, todos os nós sensores compartilham informações sobre escalonamento e o aumento do número de nós representa mais compartilhamentos. Por outro lado, no BroadLDC, o número de nós internos na árvore de difusão quando a rede tem 500, 750 ou 1000 nós é praticamente o mesmo fazendo com que o custo com descobertas e agendamentos seja similar para os três casos. No BroadLDC, o aumento do número de nós resulta no aumento do número de folhas.

A figura 8-c mostra o consumo de energia dos nós sensores. O *Gossip* faz com que eles consumam em média 1,54 e 1,74 vezes mais energia que o OF e BroadLDC, respectivamente. O consumo de energia do OF é 1,13 vezes maior que o do BroadLDC. Esses resultados são devido ao número de transmissões dos protocolos avaliados. O consumo do *Gossip* quando a rede tem 250 nós é 1,32 vezes maior que quando ela tem 1000 nós. Isso porque o aumento do número de nós aumenta a porcentagem de folhas cujo consumo de energia é menor que o dos nós internos. Pelo mesmo motivo, o consumo do BroadLDC quando a rede tem 250 nós é 1,03 vezes maior que quando ela tem 1000 nós.

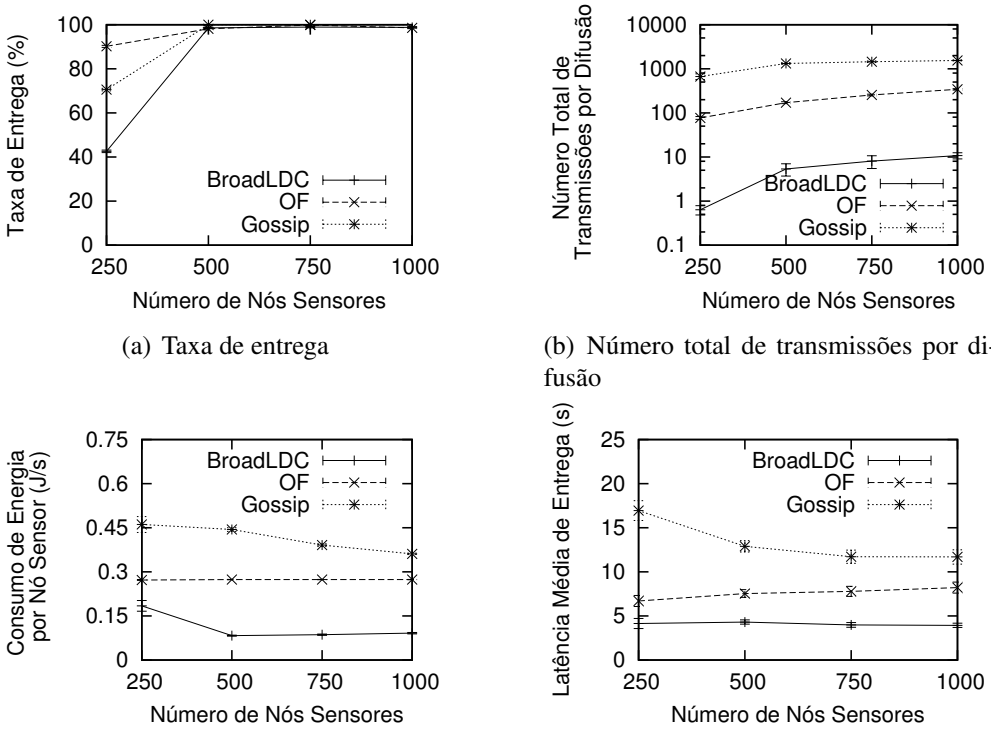
A figura 8-d mostra a latência média de entrega. A latência do *Gossip* é em média 1,77 vezes maior que a do OF e essa é 1,74 vezes maior que a do BroadLDC. Isso porque todas as propagações do *Gossip* são baseadas em agendamentos, fazendo com que cada nó emissor aguarde um tempo para decidir se propaga um pacote de difusão recebido. O resultado do OF é porque cada nó sensor conhece apenas o escalonamento de uma fração de seus vizinhos, aumentando o caminho médio entre a origem e os nós sensores. A latência do *Gossip* na rede com 250 nós é 1,36 vezes maior que na com 1000 nós. No BroadLDC, essa relação é de 1,32 vezes. Esses resultados acontecem porque o aumento o número de nós faz com que os nós emissores fiquem normalmente mais distantes entre si, aumentando a velocidade da difusão. O OF apresenta um resultado oposto em que a latência da rede com 1000 nós é 1,39 vezes maior que a com 250 nós porque o aumento do número de nós implica na redução da fração de compartilhamento do OF.

Resultados de simulação revelam que no cenário avaliado, o BroadLDC faz menos transmissões, consome menos energia e tem uma latência menor que a dos demais protocolos avaliados. Isso mostra que as soluções para difusão de dados em RSSFs com ciclo de trabalho reduzido nem sempre precisam que todos os nós compartilhem informações.

4.3. Difusão de Dados com Regiões de Baixa Energia

Esta seção avalia o BroadLDC em um cenário de difusão de dados contendo regiões de baixa energia. As figuras 9 e 10 apresentam o desempenho dos protocolos avaliados para os nós localizados dentro das regiões de baixa energia à medida que se aumenta o número de nós sensores no campo de sensoriamento. A diferença entre as duas figuras é que a primeira considera trinta e duas regiões de baixa energia e a segunda, somente uma região. Esta seção não apresenta os resultados de simulação para os nós localizados fora dessas regiões porque eles são similares aos do cenário sem regiões de baixa energia.

As figuras 9-a e 10-a apresentam a taxa de entrega para os nós localizados dentro das regiões de baixa energia. Como no cenário anterior, todos os protocolos têm a menor taxa de entrega quando a rede tem 250 nós. Na figura 9-a, as taxas de entregas do OF e a do *Gossip* foram em média 1,04 vezes maiores que a do BroadLDC quando a rede tem 250 nós. A partir de 500 nós, os três protocolos apresentam praticamente as mesmas taxas o que pode ser confirmado aplicando o *teste-T* com $0,05$ de significância. Na figura 10-a, as taxas de entrega do OF e *Gossip* foram em média 1,93 vezes maiores que a do BroadLDC. Esses resultados são porque o BroadLDC proibi que os nós localizados dentro das regiões de baixa energia sejam nós internos. A taxa de entrega média do BroadLDC quando a rede tem trinta e duas regiões é 1,85 vezes maior que quando ela tem uma região porque no primeiro cenário, as regiões têm raios menores permitindo que mais nós localizados dentro dessas regiões sejam folhas na árvore de difusão.



(a) Taxa de entrega

(b) Número total de transmissões por difusão

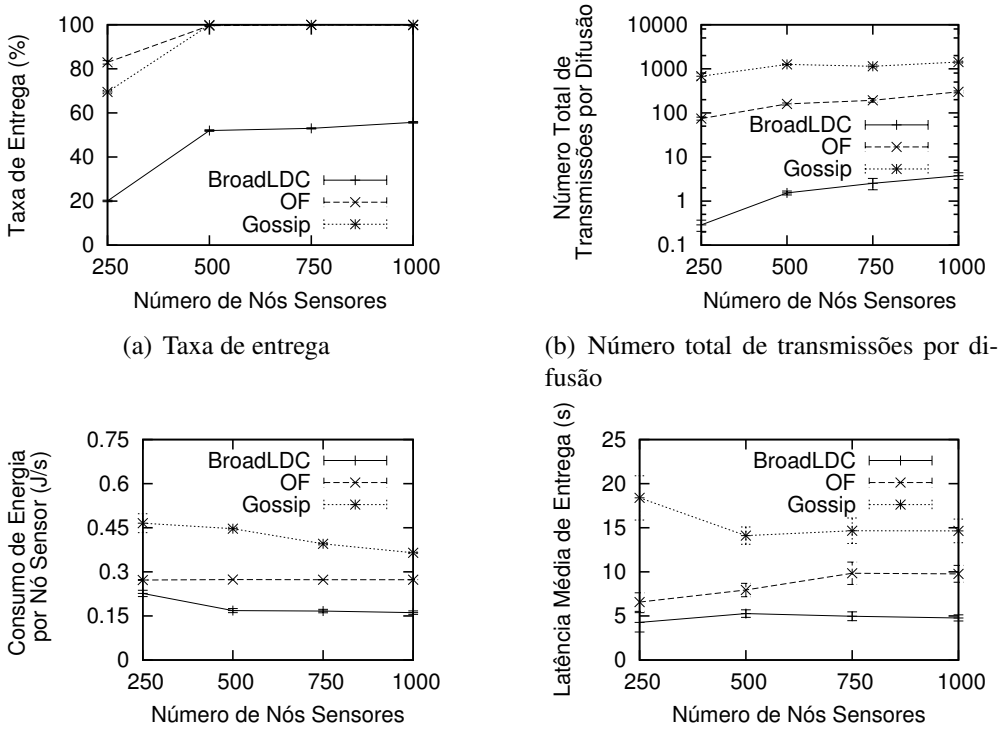
(c) Consumo de energia por nó sensor

(d) Latência média de entrega

Figura 9. Métricas avaliadas para os nós localizados dentro de trinta e duas regiões

As figuras 9-b e 10-b mostram o número total de transmissões para os nós localizados dentro das regiões de baixa energia. As figuras têm escala logarítmica no eixo y e o aumento do número de nós no campo de sensoriamento faz com que os três protocolos avaliados façam mais transmissões. Isso acontece porque tem-se um aumento na densidade da rede, fazendo com que cada região de baixa energia tenha mais nós sensores (destinos das difusões). O BroadLDC faz menos transmissões que os demais protocolos avaliados porque ele proíbe que os nós localizados dentro das regiões de baixa energia sejam nós internos. As transmissões realizadas por esse protocolo são de pacotes de filiação. Quando a rede tem trinta e duas regiões de baixa energia, o OF e *Gossip* fazem em média 34 e 201 vezes mais transmissões que o BroadLDC, respectivamente. Quando ela tem uma região, eles fazem em média 90 e 555 vezes mais transmissões que o BroadLDC, respectivamente. Na figura 9-b, o *Gossip* faz em média 6.20 vezes mais transmissões que o OF e, na figura 10-b, essa relação é de 5,89 vezes. Esse resultado é consequência dos agendamentos do *Gossip*. Comparando as figuras 9-b e 10-b, o BroadLDC faz 3,06 vezes mais transmissões no cenário com trinta e duas regiões que no com uma região. Esse resultado é porque o raio das trinta e duas regiões é menor que o da região única fazendo com que no primeiro caso, os nós localizados dentro das regiões tenham mais vizinhos externos às mesmas. Como esses vizinhos podem ser nós internos, aumenta-se o número de nós folhas e de transmissões de pacotes de filiação dentro das regiões.

As figuras 9-c e 10-c apresentam o consumo de energia dos nós sensores. O consumo de energia do OF e do *Gossip* é em média 1,99 e 3,02 vezes maior que o do BroadLDC, respectivamente, porque o BroadLDC reduz o ciclo de trabalho dos nós loca-



(a) Taxa de entrega

(b) Número total de transmissões por difusão

(c) Consumo de energia por nó sensor

(d) Latência média de entrega

Figura 10. Métricas avaliadas para os nós localizados dentro de uma região

lizados dentro das regiões com menos energia e efetua menos transmissões que os demais protocolos. Comparando as figuras 9-c e 10-c, o BroadLDC consome 1,62 vezes mais energia no cenário com uma região que no com trinta e duas. Esse resultado é porque com trinta e duas regiões, o BroadLDC faz com que mais nós localizados dentro dessas regiões participem da árvore (como folhas), reduzindo seus respectivos ciclos de trabalho.

As figuras 9-d e 10-d mostram a latência média de entrega para os nós localizados dentro das regiões. Como no cenário anterior, o *Gossip* apresenta a maior latência seguido do OF. As latências do *Gossip* e do OF são em média 3,23 e 1,81 vezes maiores que a do BroadLDC devido aos agendamentos do *Gossip* e à fração de compartilhamento do OF.

Resultados de simulação revelam que para o cenário avaliado, o BroadLDC reduz o número de transmissões, consumo de energia e latência dentro das regiões de baixa energia em relação aos demais protocolos avaliados. Ele também consegue manter a taxa de entrega quando essas regiões têm um raio menor.

5. Conclusões e Trabalhos Futuros

Este trabalho propõe e avalia o protocolo BroadLDC para a difusão de dados em RSSFs com ciclo de trabalho reduzido. A difusão de dados é uma tarefa desafiadora nesse tipo de cenário porque quando um nó sensor envia um pacote, nem todos seus vizinhos estão acordados. A ideia principal do BroadLDC é criar uma árvore de difusão baseada no mapa de energia da rede e fazer com que cada nó sensor compartilhe informações com apenas um nó vizinho que participa da árvore disseminando pacotes. O BroadLDC também altera o funcionamento básico dos nós localizados dentro das regiões de baixa energia para pro-

longar o tempo de vida da rede. Resultados de simulação mostram que o BroadLDC reduz o número de transmissões, consumo de energia e latência quando comparado com outras soluções propostas na literatura e avaliadas neste trabalho. Ele também evita regiões de baixa energia, reduzindo o fluxo de dados e o consumo de energia dentro das mesmas.

Trabalho futuros podem usar a árvore do BroadLDC para a coleta de dados. Nesse caso, quando um nó sensor tiver algum dado para o nó monitor, esse nó sensor propaga para seu pai que repete o processo até o nó monitor. Esses nós também pode explorar alguma técnica de fusão de dados. Além disso, a árvore também pode ser explorada por protocolos de clusterização em que os lideres dos *clusters* são os nós da árvore.

Referências

(2013). Mts/mda sensor and data acquisition boards user's manual, document 7430-0020-03. www.xbow.com.

Arshad, U. N. M., Shahid, N., and Raza, S. (2012). Classification of localization algorithms for wireless sensor network: A survey. In *Open Source Systems and Technologies (ICOSST), 2012 International Conference on*, pages 1–5.

Guidoni, D., Machado, M., Mini, R., and Loureiro, A. (2006). Difusão de dados baseada em atraso e energia para redes de sensores sem fio. In *Proceedings of the Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 134–143.

Guo, S., He, L., Gu, Y., Jiang, B., and He, T. (2013). Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. *IEEE Transactions on Computers*, PP(99):1–14.

Guo, S., Kim, S., Zhu, T., Gu, Y., and He, T. (2011). Correlated flooding in low-duty-cycle wireless sensor networks. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, pages 383–392.

Haas, Z., Halpern, J., and Li, L. (2006). Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14(3):479–491.

Heissenbuttel, M., Braun, T., Walchli, M., and Bernoulli, T. (2006). Optimized stateless broadcasting in wireless multi-hop networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–12.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation and modeling*.

Machado, M., Mini, R., and Loureiro, A. (2012). Política híbrida para determinar o momento acordado do próximo nó na comunicação em redes de sensores sem fio. In *Proceedings of the Simpósio Brasileiro de Redes de Computadores e Sistemas Distribídos (SBRC)*, pages 205–218.

Mini, R., Machado, M., Loureiro, A., and Nath, B. (2005). Prediction-based energy map for wireless sensor networks. *Ad Hoc Networks Journal*, 3(2):235–253.

Wang, Y., Vuran, M., and Goddard, S. (2012). Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 20(1):305–318.

Processamento Distribuído da Junção Espacial de Múltiplas Bases de Dados - *Multi-way Spatial Join*

Anderson R. Cunha[1], **Sávio S. T. de Oliveira**[1], **Everton L. Aleixo**[2], **Marcelo de C. Cardoso**[2], **Thiago B. de Oliveira**[1], **Vagner J. do Sacramento Rodrigues**[2]

[1]Instituto de Informática - INF – Universidade Federal de Goiás - UFG
Bloco IMF I - Campus II – Samambaia – Caixa Postal 131 – CEP 74001-970 - Goiânia – GO

[2]Departamento de P&D – goGeo.io
Alameda Leopoldo de Bulhões, Número 30 - Setor Pedro – Goiânia – GO

{andersoncunha,savioteles,thborges}@ufg.br

{everton.lima,marcelo.cardoso,vagner}@gogeo.io

Abstract. *This paper proposes the Distributed Synchronous Traversal algorithm (DST) for distributed processing of spatial join with multiple datasets (Multi-way Spatial Join). It was evalueted over clusters up to eigth servers, and just like the Synchronous traversal algorithm, performs spatial join by synchronously traversing the input R-Trees. The experiments results on real datasets have shown that the algorithm provides a satisfactory degree of horizontal scalability.*

Resumo. *Neste trabalho é proposto o algoritmo Distributed Synchronous Traversal (DST) para o processamento distribuído da junção espacial com múltiplas bases de dados (Multi-way Spatial Join). Ele foi avaliado sobre clusters de até oito servidores, e assim como o algoritmo Synchronous Traversal (ST), realiza o processamento da junção espacial percorrendo de forma síncrona os índices R-Tree das bases de dados de entrada. Os resultados gerados mostraram que o algoritmo apresentou um grau de escalabilidade horizontal satisfatório.*

1. Introdução

A junção espacial (*Spatial Join*) é uma das principais operações espaciais para oferta de serviços em aplicações de *Location Based Services* (LBS), e também é uma das que mais demanda recurso computacional [Mutenda and Kitsuregawa 1999] para cruzamento de duas bases de dados georreferenciadas. Sua complexidade e custo computacional aumentam significativamente quando a junção envolve múltiplas bases de dados (*multi-way spatial join*) sobre bases de dados com grande volume de dados georreferenciados (*spatial big data*).

Através de operações de *multi-way spatial join* é possível responder perguntas complexas e de alto impacto cruzando diversas bases de dados simultaneamente. Por exemplo, quais desmatamentos (base de desmatamento) no Brasil estão a 200 metros de rios (base de hidrografia) a 500 metros de áreas de preservação ambiental (base de preservação), 1000 metros de rodovias (base de rodovias) e estão em biomas do cerrado ou mata atlântica (base de biomas)?

Estratégias tradicionais de processamento do *multi-way spatial join* geralmente aplicam combinações de algoritmos de junção espacial binária (*pairwise join algorithms*)

sobre ambientes computacionais centralizados. Em consultas complexas ou sobre grandes volumes de dados, esse tipo de abordagem exige servidores com grande capacidade computacional, muitas vezes inviáveis técnica e financeiramente em ambientes centralizados.

Neste trabalho é proposto um algoritmo distribuído para processamento de operações de *multi-way spatial join* em um *cluster* de computadores. O algoritmo é chamado *Distributed Synchronous Traversal* (DST), se baseia no algoritmo *Synchronous Traversal* (ST) [Papadias et al. 1998, Papadias et al. 1999, Mamoulis and Papadias 2001] e processa o *multi-way spatial join* percorrendo de forma síncrona os índices espaciais de entrada sem geração de bases de dados intermediárias (i.e., o resultado é construído conforme os níveis são percorridos, sem etapas adicionais de persistência e/ou reindexação).

O DST foi implementado e avaliado através de experimentos sobre *clusters* com até 8 servidores em função de dois fatores importantes na execução distribuída do *multi-way spatial join*: ***i)*** a capacidade de escalabilidade horizontal; ***ii)*** a distribuição de carga do processamento.

Até onde conhecemos, não há outras propostas na literatura de algoritmo baseado em uma plataforma *peer-to-peer* para o processamento distribuído de operações de *multi-way spatial join*. Nesse contexto, podemos resumir as contribuições deste trabalho da seguinte forma:

- Projeto e implementação de um algoritmo distribuído para processamento do *multi-way spatial join* de forma escalável em um *cluster* de computadores;
  - Definição de uma estrutura para mapeamento da semântica da consulta e coordenação do processamento distribuído da operação;
  - Algoritmos de restrição espacial para diminuir o número total de operações de junção espaciais binárias e o tráfego de dados na rede.
- Avaliação e experimentação do DST segundo diferentes critérios de análise com bases de dados reais.

2. Processamento de Dados Espaciais

Diversas pesquisas foram realizadas para desenvolver métodos de acesso e estruturas de indexação que fossem adequadas às características dos dados espaciais (multidimensionais) [Bentley 1975, Guttman 1984, Kamel and Faloutsos 1994]. Entre estas estruturas de indexação de dados espaciais, a *R-tree* [Guttman 1984] e suas variações são as mais citadas na literatura e utilizadas em bancos de dados geográficos, tais como Oracle Spatial, PostGIS e MySQL Spatial.

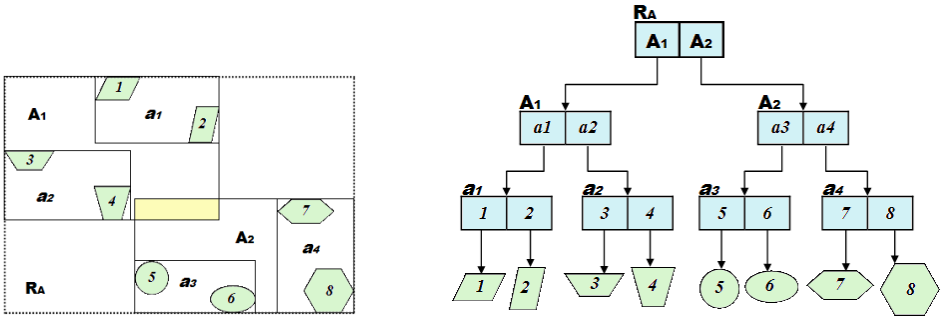
2.1. *Estrutura de Indexação (R-Tree)*

A *R-Tree* é uma estrutura hierárquica que utiliza retângulos para organizar objetos espaciais de forma que objetos colocalizados (i.e., espacialmente próximos) fiquem armazenados próximos uns dos outros, promovendo assim uma redução no espaço de busca a cada nível da árvore. A *R-Tree* é uma árvore balanceada por altura com ponteiros para os objetos espaciais nos nós folhas (ela pode ser vista como uma extensão da B^+*-Tree* [Comer 1979] no espaço multidimensional).

Os retângulos utilizados pelas *R-Trees* são denominados de *Minimum Bounding Rectangles* (MBRs) . Um MBR possui a menor área retangular possível para englobar

completamente seus objetos espaciais. A Figura 1(a) retrata o desenho dos MBRs agrupando os objetos espaciais de 1 a 8 em subconjuntos, de acordo com sua localização espacial.

A Figura 1(b) ilustra a estrutura hierárquica da *R-Tree* apresentando um nó raiz (R_A), nós diretórios (A_1 e A_2) e um último nível com nós folhas (a_1 até a_4). Cada nó armazena no máximo M e no mínimo $m \leq \frac{M}{2}$ entradas. Entre as várias extensões propostas para a *R-Tree*, a R*-Tree propõe mecanismos para reduzir a área morta e as áreas de sobreposição entre os MBRs melhorando o tempo de busca em bases de dados dinâmicas [Beckmann et al. 1990].



(a) Disposição espacial dos objetos geográficos

(b) Índice espacial *R-Tree*

Figura 1. *R-Tree*, representações e estruturas

2.2. *Multi-way Spatial Join*

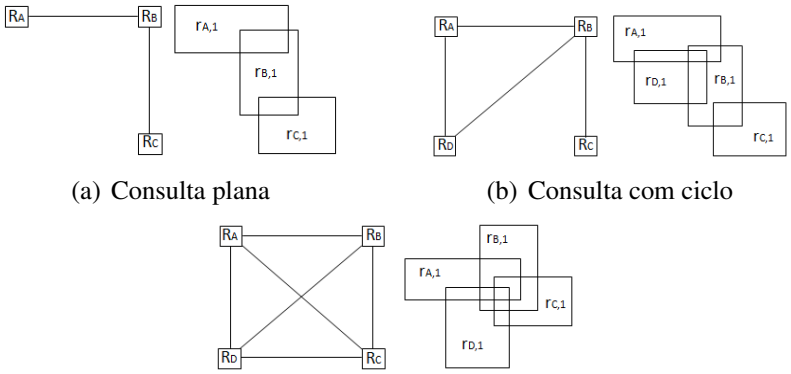
A junção espacial sobre um número arbitrário de bases de dados (*multi-way spatial join*) pode ser formalmente definida da seguinte maneira: dado um conjunto de *n* bases de dados $R_1, R_2, ..., R_n$ e uma consulta Q, onde Q_{ij} é o predicado espacial que deve ser aplicado entre R_i e R_j, retornar todas as *n-uplas* $\{(r_{1,w}, ..., r_{i,x}, ...r_{j,y}, ...r_{n,z}) \,|\, \forall i, j \,:\, r_{i,x} \in R_i,\, r_{j,y} \in R_j \text{ e } r_{i,x}\, Q_{ij}\, r_{j,y}\}$ [Mamoulis and Papadias 2001]. Neste trabalho, a operação de intersecção de geometrias foi utilizada como predicado espacial Q_{ij}.

A consulta Q pode ser vista como um grafo $G = (N, E)$, onde *N* representa o conjunto de bases de dados de entrada, e *E* representa os predicados de junção, assim como em uma rede de restrições (*constraint network*) [Papadias et al. 1999], onde os vértices correspondem as variáveis do problema, e as arestas a restrições espaciais binárias. Essa rede de restrições foi denominada Grafo de Consulta e apresentado em maiores detalhes a seguir.

A Figura 2 apresenta exemplos de consultas *Q*, e possíveis soluções representadas por retângulos. A Figura 2(a) apresenta uma consulta plana com as restrições: $r_{A,1} \cap r_{B,1} \neq \emptyset$ e $r_{B,1} \cap r_{C,1} \neq \emptyset$. A Figura 2(b) apresenta uma consulta com ciclo com as restrições: $r_{A,1} \cap r_{B,1} \neq \emptyset$, $r_{B,1} \cap r_{C,1} \neq \emptyset$, $r_{B,1} \cap r_{D,1} \neq \emptyset$ e $r_{A,1} \cap r_{D,1} \neq \emptyset$. E, por fim, a Figura 2(c) apresenta uma consulta completa (clique), com a restrição: $r_{A,1} \cap r_{B,1} \cap r_{C,1} \cap r_{D,1} \neq \emptyset$.

2.3. Algoritmo *synchronous traversal* (ST)

O algoritmo *Synchronous Traversal* [Papadias et al. 1998, Papadias et al. 1999, Mamoulis and Papadias 2001], que serviu de base para a implementação do DST,



(a) Consulta plana (b) Consulta com ciclo

(c) Consulta clique

Figura 2. Exemplos de Consultas com múltiplas bases de dados

processa o *Multi-way Spatial Join* através da análise de todos os índices de entrada, seguindo combinações de nós que satisfazem as restrições de consulta, de forma similar ao que acontece com o RJ (*R-Tree Join*) [Brinkhoff et al. 1993].

Considere o exemplo de análise de múltiplas bases de dados espaciais, onde um ambientalista pretende identificar possíveis focos de assoreamento em rios que passam por áreas de conservação ambiental. Para tanto, ele tem que fazer a correlação entre as bases de dados espaciais de Desmatamento, Hidrografia e Áreas de conservação ambiental da região de interesse.

Essas bases de dados e os seus respectivos índices *R-Tree* são apresentados na Figura 3. Neste exemplo, a analise *multi-way spatial join* pode ser mapeada em uma consulta plana (como a consulta mostrada na Figura 2(a)) que busca pelo conjunto de todas as 3-uplas (ou triplas) de desmatamento, rios e áreas de conservação que atendem a seguinte restrição: desmatamentos que intersectam com rios que passam por áreas de conservação ambiental.

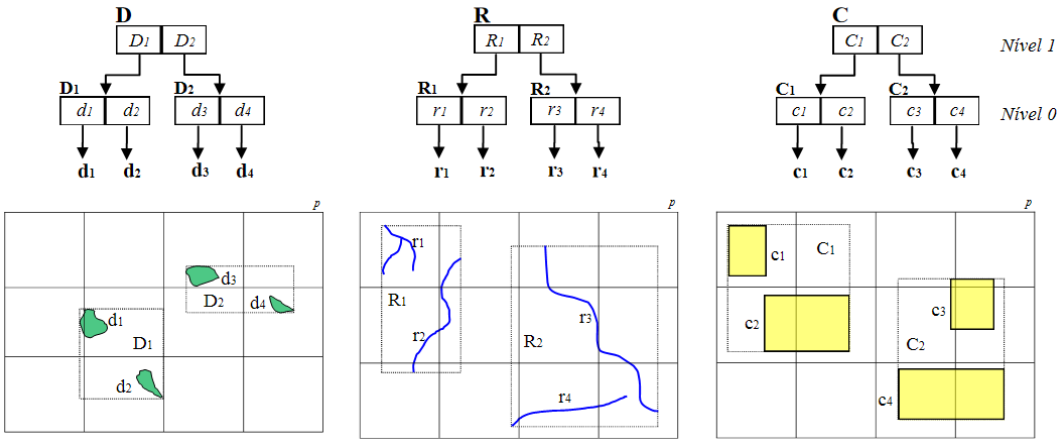


Figura 3. Representação gráfica das bases de dados Desmatamento, Rios e Áreas de Conservação sobre o plano cartesiano, e os respectivos índices R-TRee.

ST realiza o processamento da junção espacial em duas fases, filtragem e refina-

mento: Na fase de filtragem, o ST inicia o processamento nas raízes das *R-Trees*. Onde resolve um único subproblema, ou seja, checagem da combinação dos nós raízes das *R-Tree* a fim de encontrar as candidatas a soluções para processamento no próximo nível. Este processo se repete recursivamente através dos níveis intermediários até que o nível das folhas (nível 0) seja atingido, e se inicie a fase de refinamento com o processamento do resultado final da junção.

No exemplo, das oito combinações possíveis no nível das raízes das *R-Trees* (i.e., $(D_1, R_1, C_1), (D_1, R_1, C_2), ..., (D_2, R_2, C_2)$), apenas $(D_1, R_1, C_1), (D_1, R_2, C_2)$ e (D_2, R_2, C_2) podem levar a solução final e constituem o resultado da fase de filtragem. As demais combinações são descartadas por não atenderem às restrições da consulta.

Na fase de refinamento, o processamento das combinações formadas pelos filhos da candidata a solução (D_1, R_1, C_1) leva à tripla (d_1, r_2, c_2) que faz parte do resultado final. A candidata a solução (D_2, R_2, C_2) leva a outras duas triplas (d_3, r_3, c_3) e (d_3, r_3, c_4) que completam o resultado final da consulta (a tripla (D_1, R_2, C_2) é um falso positivo).

3. Processamento Distribuído da Junção Espacial com Múltiplas Bases de Dados

Para o processamento distribuído do DST foram utilizados servidores em um cluster onde se encontra um índice espacial *R-Tree* Distribuído [de Oliveira et al. 2013], conforme Figura 4, que aliado a mecanismos para o tratamento de concorrência, consistência e um serviço de descoberta possibilitam a paralelização e distribuição do processamento de operações espaciais.

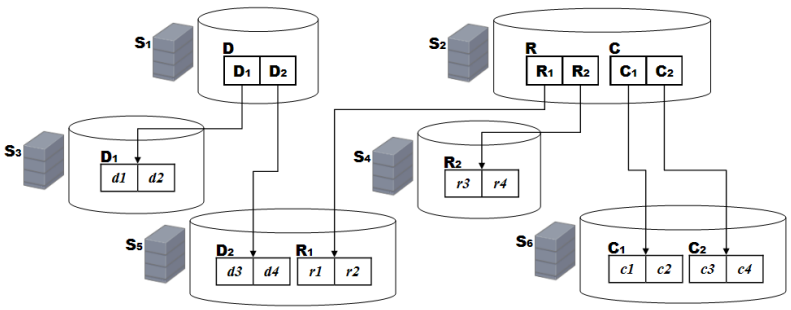


Figura 4. *R-Trees* das bases de dados da Figura 3 distribuídas entre 6 Servidores

As requisições de operações espaciais são encaminhadas para o servidor do *cluster* que contém os dados. O servidor que recebe a requisição necessita, eventualmente, buscar dados em outros servidores ou mesmo repassar para outro servidor a requisição da operação. A troca de dados entre os servidores é minimizada através da escolha da estratégia de distribuição de dados. Neste trabalho, foram experimentados dois algoritmos de distribuição conforme descrito na Seção 4.1.

3.1. *Distributed Synchronous Traversal* (DST)

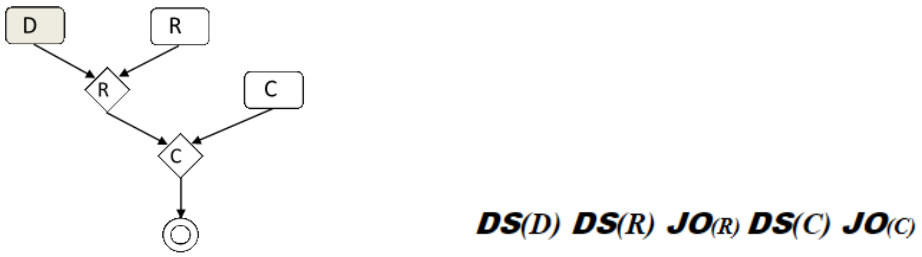
O algoritmo DST adota a abordagem de busca sistemática (Systematic Search Algorithms), como os algoritmos de *backtracking* utilizados em problemas de satisfação de restrições binárias (*binary cosntraint satisfaction problem* - BCSP) [Prosser 1993]. Segundo Bacchus e Van Beek [Bacchus and van Beek 1998], um problema BCSP é definido

sobre um conjunto finito de variáveis n, cada uma com um domínio Δ_i também finito de valores potenciais e uma coleção finita de restrições.

O problema de multi-way spatial join pode ser reduzido para um problema BCSP, uma vez que o conjunto de variáveis n vai ser composto pelos nós das *R-Tree* de entrada, os domínios Δ_i são os filhos dos respectivos nós das *R-Trees* e as restrições binárias C_{ij} entre os pares de variáveis (v_i, v_j) são as arestas do grafo de consulta.

Logo o DST, assim como os algoritmos de *backtracking*, processa a consulta nível a nível. O conjunto de soluções em níveis intermediários das *R-Trees* definem os parâmetros para novos problemas BCSP nos níveis inferiores, que serão processados recursivamente até que no nível das folhas. As soluções de todos os problemas BCSP compõem o resultado final da consulta e são retornados (A Figura 6 ilustra esse comportamento).

Esse processamento realizado pelo DST é semelhante ao processamento realizado pelo ST (vide Seção 2.3), só que de forma distribuída sobre um *cluster* de computadores, conforme ilustrado na Figura 4. Para distribuir o processamento do *multi-way spatial join* pelas máquinas do cluster foi necessário definir um Plano de Consulta que viabilizasse a criação de um Plano de Execução que pode ser executado de forma coordenada e distribuída sobre o *cluster* de computadores.



(a) Plano de Consulta (b) Plano de Execução

Figura 5. Exemplo de Plano de Consulta e Plano de Execução resultante.

A consulta Q deve ser fornecida para o algoritmo na forma de uma árvore binária[1] que por convenção recebe o nome de Plano de Consulta (Figura 5(a)). Esta árvore é percorrida em um percurso de pós-ordem para gerar o Plano de Execução que é uma estrutura baseada em expressões pós-fixas (Figura 5(b)) e será utilizado pelo DST para coordenar o processamento distribuído da consulta.

O Plano de Execução usa a seguinte notação $DS(X)$ para definir as *R-Trees* de entrada (*dataset*) e $JO_{(Y)}$ para representar as operações de junção espacial (operações), tal que X representa o nome da base de dados de entrada (retângulos da Figura 5(a)) e Y compreende os resultados da operação de junção JO (losangos da Figura 5(a)). Esse resultado da operação $JO_{(Y)}$ entra como argumento para a próxima operação de junção, e recebe o nome de $JR_{(Y)}$ (ver Figura 6).

[1] A conversão do Grafo de Consulta para o Plano de Consulta não está no escopo desse trabalho por lidar com algoritmos de otimização dos planos de consulta.

3.2. Processamento do Plano de Execução

No algoritmo DST, todas as operações de junção espacial ($JO_{(X)}$) do Plano de Execução serão executadas no servidor do *cluster* que contém os dados do primeiro argumento da operação. Esta característica aumenta o potencial de escalabilidade do sistema, pois distribui o processamento entre os vários servidores do *cluster*.

Um Plano de Execução $PE_{(k,n)}$ implica no processamento de uma n-upla n no nível k da *R-Tree*. A Figura 6 apresenta, como exemplo, a instância de processamento do Plano de Execução $PE_{(1,0)}$ (Figura 5(b)) sobre as bases de dados da Figura 3. O processamento de $PE_{(1,0)}$ irá resultar em $JR_{(C)}$ contendo as triplas $(D_1, R_1, C_1), (D_2, R_2, C_2)$ e (D_1, R_2, C_2). Essas triplas serão convertidas nos Planos de Execução $PE_{(0,0)}$, $PE_{(0,1)}$ e $PE_{(0,2)}$ que serão redistribuídos pelos servidores do *cluster*.

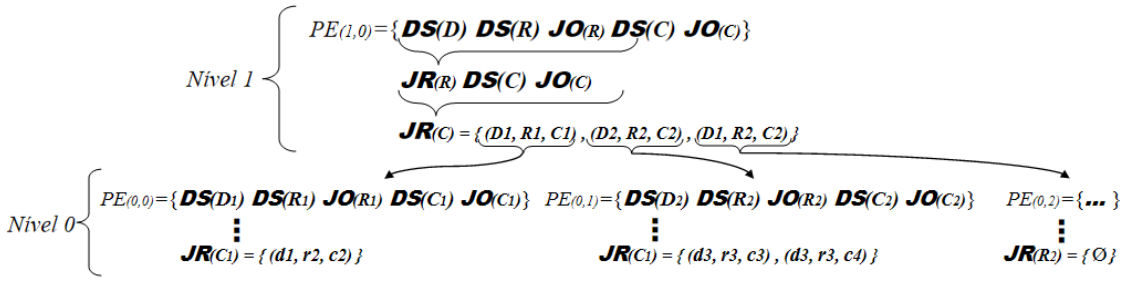


Figura 6. Instância do processamento do plano de execução da Figura 5(b)

No nível das folhas (fase de refinamento), o Plano de Execução $PE_{(0,0)}$ resulta na tripla (d_1, r_2, c_2) e o Plano de Execução $PE_{(0,1)}$ resulta nas triplas (d_3, r_3, c_3) e (d_3, r_3, c_4), estas 3 triplas são retornadas por constituirem o resultado final da consulta. O $PE_{(0,2)}$ é um falso positivo, pois não há intersecção entre as filhos dos nós D_1 e R_2 (ver Figura 3).

4. Experimentos

Os experimentos foram realizados sobre bases de dados reais fornecidas pelo LAPIG[2]. As bases de dados utilizadas e suas características são descritas a seguir: ***1)*** Bioma do Cerrado (*B*), 151986 Polígonos e 411,3MB; ***2)*** Desmatamento do Cerrado (*D*), 32578 Polígonos e 11.2MB; ***3)*** Ferrovias do Brasil (*F*), 617 Linhas e 668KB; ***4)*** Hidrografia do Brasil (*H*), 226963 Linhas e 64,5MB; ***5)*** Hidrovias do Brasil (*I*), 5571 Linhas e 1.4MB; ***6)*** Localidades (*L*), 21840 Pontos e 1,4MB; ***7)*** Municípios do Brasil (*M*), 5564 Polígonos e 38.8MB; ***8)*** Rodovias (*R*), 51645 Linhas e 15,2MB; e ***9)*** Vegetação do Brasil (*V*), 2140 Polígonos e 4.7MB.

Estas bases de dados foram combinadas em consultas planas (ver Figura 2(a)) com 3 e 4 entradas, variando o tamanho das bases de dados, a quantidade de entradas, e o tipo de geometria processada. Abaixo são descritos os planos de consulta, incluindo o Plano de Execução equivalente e o número de resultados gerados por cada uma:

- ***1-*** Hidrovias do Brasil x Hidrografia do Brasil x Rodovias do Brasil x Ferrovias do Brasil (***IHRF***, 285 resultados):
  - $DS(I)\, DS(H)\, JO_{(H)}\, DS(R)\, JO_{(R)}\, DS(F)\, JO_{(F)}$;

[2]Laboratório de Processamento de Imagens e Geoprocessamento (www.lapig.iesa.ufg.br/lapig/).

- ***2-*** Vegetação do Brasil x Desmatamento do Cerrado x Municípios do Brasil (***VDM***, 36470 resultados):
  - $DS(V)\, DS(D)\, JO_{(D)}\, DS(M)\, JO_{(M)}$;
- ***3-*** Municípios do Brasil x Localidades do Brasil x Rodovias do Brasil (***MLR***, 507556 resultados):
  - $DS(M)\, DS(L)\, JO_{(M)}\, DS(R)\, JO_{(R)}$.

4.1. Metodologia

Foram definidas 4 configurações de *cluster* para a realização dos experimentos: ***cluster1*** com 1 servidor, ***cluster2*** com 2 servidores, ***cluster4*** com 4 servidores e ***cluster8*** com 8 servidores. Cada servidor é uma instância de máquina virtual (MV) com 4 vCPUs de 2.4 Ghz, 8 GB de memória RAM, 12 GB de disco e com interface de rede 1 Gbit, com o SO Ubuntu 12.04 LTS.

Foram realizadas centenas de consulta e utilizadas na avaliação um total de 140 resultados de consultas de *multi-way spatial join* sobre diferentes configurações do sistema (i.e., combinações de Plano de Consulta, distribuição de dados, número de servidores e aplicação ou não do DSR). A fim de determinar qual o algoritmo de distribuição de dados se adequa melhor ao DST, o plano de consulta ***VDM*** foi processado sobre todas as configurações de *cluster*, utilizando as 2 distribuições de dados avaliadas, totalizando 40 consultas (Seção 4.2).

Para avaliar o desempenho e o grau de escalabilidade do DST os planos de consulta ***IHRF***, ***VDM*** e ***MLR*** foram processados sobre as 4 configurações de *cluster* somando mais 60 execuções ao total de consultas realizadas (5 execuções x 3 planos de consulta x 4 configurações de *cluster*). Os resultados são apresentados e discutidos na Seção 4.2.

Em cada uma dessas execuções foram coletadas as seguintes métricas: utilização média de CPU; média de kbytes trafegados na rede; quantidade de mensagens trocadas; tempo de resposta; quantidade de operações de junção binária; quantidade de planos de execução processados na fase de filtragem; quantidade de candidatos à solução da fase de filtragem e quantidade de falsos positivos na fase de refinamento.

Para viabilizar a experimentação do algoritmo DST foi utilizada a plataforma de middleware de geoprocessamento distribuído *DistGeo* [de Oliveira et al. 2013] que conta com uma arquitetura de comunicação baseada no modelo *peer-to-peer*, onde os servidores do cluster trocam informações de descoberta utilizando o protocolo *Gossip* [Demers et al. 1987], além de recursos interessantes como diferentes formas de distribuição de dados e o tratamento de concorrência, consistência no processamento de operações espaciais.

Cada uma das consultas avaliadas nos experimentos também foram processada em uma instância do PostGIS com o propósito de validar os resultados obtidos pelo DST que retornou os mesmos resultados que o PostGIS conforme esperado.

4.2. Avaliação de desempenho do algoritmo DST

Foram realizados experimentos utilizando os dois algoritmos de distribuição de dados *Round-robin* [de Oliveira et al. 2011] e *Grid-proximity* [de Oliveira 2013], a fim de selecionar aquele que melhor atende às particularidades do DST.

A distribuição *Grid-proximity-area* alcança o seu principal objetivo que é reduzir o tráfego de dados na rede. Por outro lado, ele gera gargalos de processamento devido à aglomeração de dados espacialmente próximos em um pequeno número de servidores. Com a utilização do *Round-robin*, graças à distribuição natural dos dados entre os servidores do *cluster*, apesar da geração de tráfego de dados um pouco maior (em alguns casos chegou a ser 9 vezes maior), a melhor distribuição dos dados e carga de processamento levou a um melhor aproveitamento dos recursos do *cluster*.

Em função do comportamento observados na análise dos algoritmos de distribuição, os demais resultados de avaliação de desempenho apresentados nessa seção foram obtidos com a utilização da distribuição de dados *Round-robin*, que se mostrou mais adequada ao algoritmo DST.

4.2.1. Plano de Consulta *1- IHRF*

A consulta IHRF é computacionalmente cara, devido ao grande número de operações de junção espacial binária (202.536.882 junções) e pela alta porcentagem de falsos positivos na fase de refinamento, aproximadamente 81% dos candidatos a solução. Isso se deve pela grande quantidade de espaço morto gerado pelos MBRs dos índices *R-Tree* das tabelas com geometrias do tipo Linha. Apesar do alto custo computacional dessa consulta, o DST conseguiu alcançar um bom grau de escalabilidade.

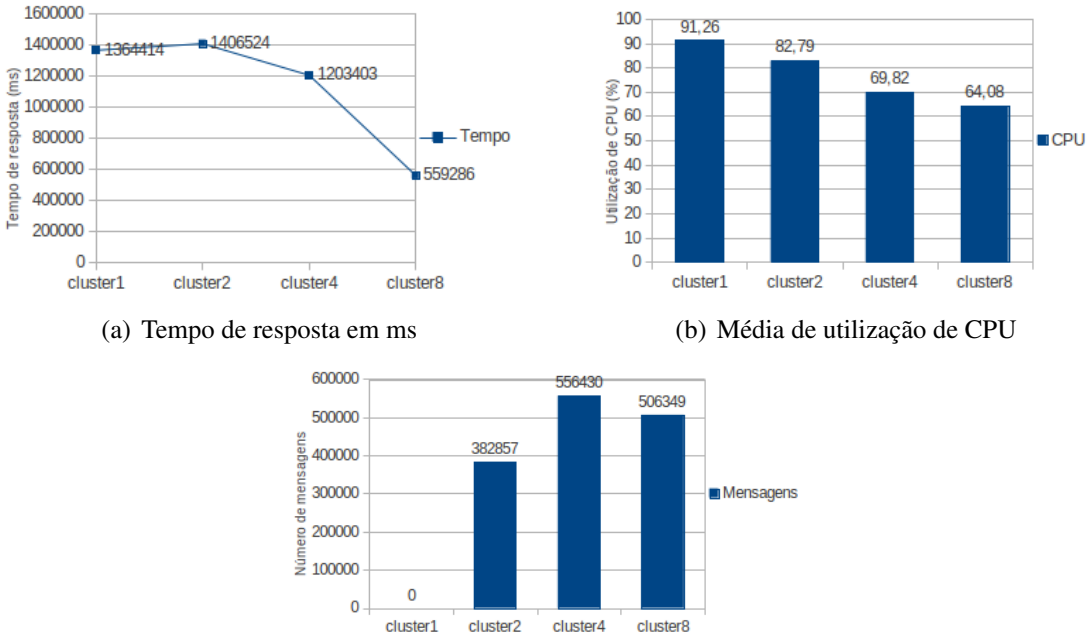
Entre os *clusters* com 1 e 2 servidores houve uma pequena piora de aproximadamente 3% no tempo de resposta, devido a introdução do tráfego de dados na rede (que não ocorre no ***cluster1***). A partir do cluster2 o tempo de resposta melhorou aproximadamente 16,9% no cluster4 e aproximadamente 115,17% no cluster8. Isto indica uma melhora progressiva no tempo de resposta a partir do ***cluster2*** conforme mostrado pela Figura 7(a).

O principal fator que contribuiu com o bom resultado obtido na consulta ***IHRF*** sobre o ***cluster8*** foi o alto grau de colocalização dos dados, evidenciada pela Figura 7(c). Essa colocalização foi gerada ao acaso, uma vez que o algoritmo de distribuição *Round-robin* não tem nenhum controle sobre a colocalização dos dados. Entretanto, este fenômeno favoreceu o desempenho do DST que não precisa trocar mensagens na rede para processar os dados colocalizados.

Por ter sido reduzido o tráfego de dados na rede, foi possível obter, em todas as configurações de *cluster*, uma porcentagem alta e uniforme de utilização de CPU, como pode ser observado na Figura 7(b). Esta alta porcentagem de utilização foi a responsável pelos fatores de escalabilidade horizontal obtidos.

4.2.2. Plano de Consulta *2- VDM*

A Figura 8(a) apresenta a escalabilidade do tempo de resposta da consulta ***VDM*** sobre as diferentes configurações de *cluster*. A Figura 8(a) mostra uma redução de aproximadamente 6,5% no tempo de resposta do ***cluster1*** para o ***cluster2***. Essa porcentagem de melhora discreta se deve à comunicação dos servidores pela rede no ***cluster2***, que provocou a troca de 29393 mensagens no **cluster2** (vide Figura 8(b)). No entanto, a partir do



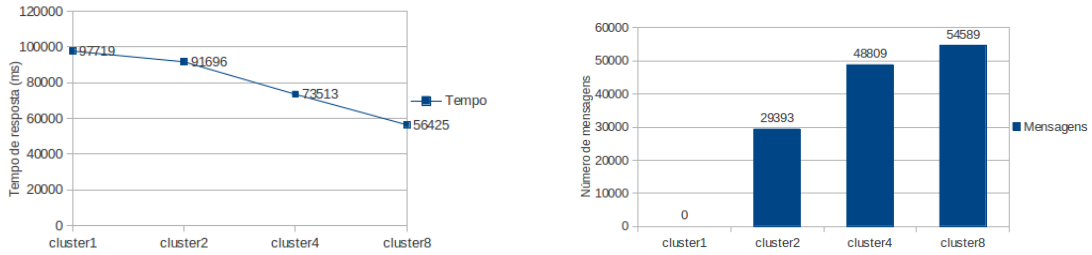
(a) Tempo de resposta em ms

(b) Média de utilização de CPU

(c) Número de mensagens trocadas

Figura 7. Consulta 1- Hidrovias X Hidrografia X Rodovias X Ferrovias.

cluster2, o fator de escalabilidade no tempo de resposta foi mais significativo, apresentando uma redução no tempo de resposta de 19,83% no ***cluster4*** em relação ao ***cluster2***, e de 23,24% no ***cluster8*** em relação ao ***cluster4***.



(a) Tempo de resposta em ms

(b) Número de mensagens trocadas

Figura 8. Consulta 2- Vegetação x Desmatamento x Municípios.

Conforme pode ser observado na Figura 8(b), o ***cluster4*** apresentou um aumento de aproximadamente 1,6 vezes no número de mensagens em relação ao ***cluster2***, no caso do ***cluster8*** esse aumento foi menor, 1,1 vezes em relação ao ***cluster4*** apesar do número de servidores ter dobrado.

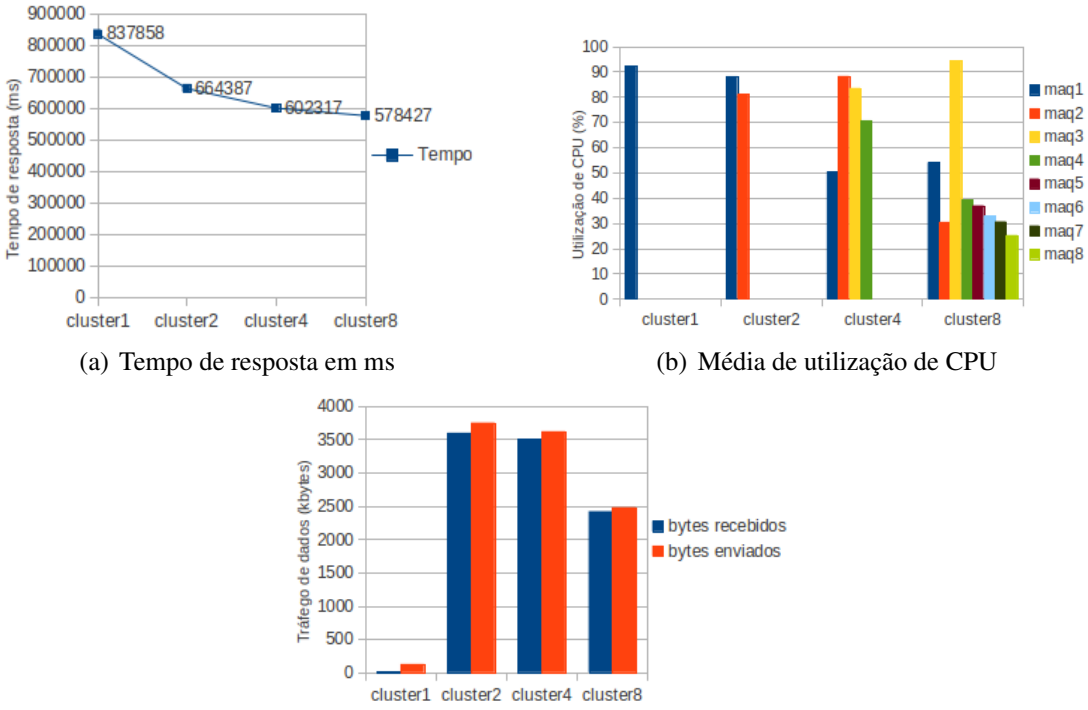
Isso indica que houve colocalização de dados das diferentes bases de dados envolvidas na consulta de forma balanceada sem a geração de gargalos de processamento (apesar dos dados espacialmente próximos estarem no mesmo servidor, os dados foram distribuídos entre os servidores de forma balanceada). Conforme discutido na Seção 4.2.1, essa colocalização foi gerada ao acaso. Isso provocou uma progressão menor no número de mensagens devido à redução da obtenção de dados na rede para o processamento da junção espacial das partes do plano de execução. Esse comportamento também ocorreu

no processamento da consulta ***IHRF***, só que em uma escala ainda maior (vide Seção 4.2.1).

A boa distribuição da carga de processamento fez com que a média de utilização de CPU em cada uma das configurações dos *clusters* reduzisse em uma taxa razoavelmente baixa. Isso garantiu uma porcentagem de utilização de CPU relativamente alta, o que possibilitou o fator de escala apresentado.

4.2.3. Plano de Consulta *3- MLR*

Na consulta ***3- MLR***, o DST também apresentou escala no tempo de resposta, mas foi observado um padrão de escala menor do que nas demais consultas, conforme ilustrado na Figura 9(a). Ao contrário do que ocorreu nas consultas ***IHRF*** e ***VDM***, o fator de escala piora conforme se acrescenta servidores no sistema. Com uma melhora de aproximadamente 20,7% do cluster1 para o cluster2, a partir do cluster2, a porcentagem de melhora reduz, caindo para 9,34% e aproximadamente 4% nas transições para *clusters* de 4 e 8 servidores.



(a) Tempo de resposta em ms

(b) Média de utilização de CPU

(c) Tráfego de dados na rede em kbytes

Figura 9. Consulta 3- Municípios x Localidades x Rodovias.

Este comportamento ocorreu porque com essa combinação houve uma tendência de formação de gargalos de processamento (i.e., poucas máquinas ficaram muito ocupadas enquanto outras ficaram ociosas). Devido à colocalização de dados em um pequeno número de servidores. A Figura 9(b) evidencia essa formação de gargalo, muito clara no ***cluster8***, onde a maioria dos servidores, teve média de uso de CPU abaixo de 30%, enquanto a média de um dos servidores foi de 90%.

A Figura 9(c) dá outra evidência ao mostrar a redução da média de dados enviados e recebidos na rede conforme se aumenta o número de servidores. Isso ocorre em parte pela colocalização dos dados, que acabou gerando o gargalo de processamento, e em parte pela característica das bases de dados, uma vez que dados geográficos do tipo ponto e linhas geram menos tráfego de dados do que polígonos [de Oliveira 2013].

5. Trabalhos Correlatos

Apesar da importância das consultas multi-way spatial join, poucos trabalhos investigaram o seu processamento em ambientes distribuídos, conforme pode-se observar na revisão da literatura feita em [Jacox and Samet 2007] e também em revisões literárias de trabalhos mais recentes como [Gupta et al. 2013]. Nosso trabalho apresenta um algoritmo distribuído para processamento de consultas multi-way spatial join, usando índices espaciais distribuídos e sem persistir resultados intermediários.

Gupta et. al. [Gupta et al. 2013] também apresenta um algoritmo para processamento distribuído de consultas multi-way spatial join. O algoritmo implementado é baseado em spatial hash join [Lo and Ravishankar 1996], que processa bases de dados não indexadas. O paradigma de programação distribuída map-reduce é empregado para o processamento distribuído da consulta. O espaço geográfico representado nas bases de dados é dividido em partes iguais utilizando uma grade, de forma que cada célula contenha uma quantidade de objetos espaciais. Cada célula é então mapeada na função map do paradigma e as células colocalizadas são correlacionadas para produzir os resultados da consulta na função reduce. Por natureza, o paradigma map-reduce persiste resultados intermediários em um sistema de arquivos distribuídos.

As diferentes abordagens entre nosso trabalho e o de [Gupta et al. 2013] tornam difícil uma comparação de tempo de processamento e uso de recursos. Porém, os tempos de processamento estão em grandezas diferentes (minutos contra horas). Apesar da diferença no tamanho das bases de dados processadas, há ainda um outro fator importante: nossos tempos incluem a etapa de refinamento, ou seja, o processamento dos objetos espaciais completos, enquanto [Gupta et al. 2013] afirmou considerar apenas uma simplificação dos objetos espaciais (MBR). A etapa de refinamento é a etapa mais demorada de uma junção espacial, conforme discutido em [Jacox and Samet 2007].

Outras abordagens usando o paradigma map-reduce foram publicadas recentemente, como [Zhang et al. 2009], e propõem técnicas para o processamento de consultas de junção espacial binária, utilizando algoritmos baseados em spatial hash-join, sem indexar as bases de dados. [de Oliveira et al. 2013] descreve um algoritmo de processamento distribuído da junção espacial binária, porém utilizando o paradigma peer-to-peer e empregando bases de dados indexadas. Todas estas técnicas, por si só, não são suficientes para o processamentos de consultas multi-way spatial join. É necessário definir como os resultados intermediários serão combinados em etapas sucessivas.

Nesta direção, [Mamoulis and Papadias 2001] apresenta um estudo extensivo de um otimizador de consultas, o qual define um plano de execução em etapas sucessivas, com uma combinação de algoritmos de junção espacial. Os autores usam histogramas de cardinalidade com duas dimensões para estimar a seletividade das junções e reduzir os resultados intermediários, evitando assim planos de execução ruins. No entanto, somente o uso de algoritmos de junção centralizados e o custo de E/S local são considerados na

definição dos planos. Seus resultados não podem ser diretamente aplicados aos algoritmos de junção distribuídos sem que antes seja estudado o impacto da distribuição dos dados no tempo de execução final da consulta.

6. Conclusões

Neste trabalho, buscou-se explorar o potencial dos sistemas distribuídos na solução de um problema complexo e com alto custo computacional, o processamento da junção espacial de um número arbitrário de bases de dados de entrada (*multi-way spatial join*).

Para tanto, muitos desafios tiveram que ser superados, a definição de um Plano de Execução que possibilitasse mapear a semântica das consultas de *multi-way spatial join* e, ao mesmo tempo, coordenar de forma distribuída o processamento das múltiplas operações de junção espacial.

O maior desafio em qualquer sistema distribuído é distribuir as tarefas o mais uniformemente possível para evitar subutilização de recursos computacionais. Os resultados demonstraram que o algoritmo DST conseguiu superar este desafio, ao se mostrar escalável em todos os experimentos realizados.

O algoritmo DST apresentou um bom grau de escalabilidade em consultas formadas por diferentes tipos de bases de dados reais, mesmo ao serem processadas através de Planos de Consulta simples, que não levam em conta na sua construção fatores como tipo ou cardinalidade das bases de dados.

Durante a confecção deste trabalho, foram identificados diversos pontos de otimização que podem melhorar o desempenho do DST. Os principais deles foram elencados abaixo:

- Definir algoritmos de otimização para a conversão automática de grafos de consulta em planos de consulta;
- Melhorar o algoritmo para paralelizar o processamento (em um mesmo nível) de partes independentes do Plano de Execução;
- Desenvolver um algoritmo de distribuição de dados mais adequado ao *multi-way spatial join*.

Referências

Bacchus, F. and van Beek, P. (1998). On the conversion between non-binary and binary constraint satisfaction problems. In *AAAI/IAAI*, pages 310–318.

Beckmann, N., Kriegel, H., Schneider, R., and Seeger, B. (1990). *The R*-tree: an efficient and robust access method for points and rectangles*, volume 19. ACM.

Bentley, J. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):517.

Brinkhoff, T., Kriegel, H., and Seeger, B. (1993). *Efficient processing of spatial joins using R-trees*, volume 22. ACM.

Comer, D. (1979). Ubiquitous B-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137.

de Oliveira, S., Sacramento, V., Cunha, A., Aleixo, E., de Oliveira, T. B., de Castro Cardoso, M., and Junior, R. R. (2013). Processamento Distribuído de Operações de Junção

Espacial com Bases de Dados Dinâmicas para Análise de Informações Geográficas. In *SBRC 2013 (XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos)*, Brasília, Brazil.

de Oliveira, S. S. T. (2013). DistJoin: Plataforma de Processamento Distribuído de Operações de Junção Espacial com Bases de Dados Dinâmicas. Mestrado, Instituto de Informática - Universidade Federal de Goiás.

de Oliveira, T., Sacramento, V., Oliveira, S., Albuquerque, P., and Cardoso, M. (2011). DSI-RTree-Um Índice R-Tree Escalável Distribuído. In *SBRC 2011 (XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos)*, Campo Grande, Brazil.

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., and Terry, D. (1987). Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12. ACM.

Gupta, H., Chawda, B., Negi, S., Faruquie, T. A., Subramaniam, L. V., and Mohania, M. (2013). Processing multi-way spatial joins on map-reduce. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 113–124. ACM.

Guttman, A. (1984). *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM.

Jacox, E. H. and Samet, H. (2007). Spatial join techniques. *ACM Transactions on Database Systems (TODS)*, 32(1):7.

Kamel, I. and Faloutsos, C. (1994). Hilbert R-tree: An Improved R-tree using Fractals. In *VLDB 20th*, page 509. Morgan Kaufmann Publishers Inc.

Lo, M.-L. and Ravishankar, C. V. (1996). Spatial hash-joins. In *ACM SIGMOD Record*, volume 25, pages 247–258. ACM.

Mamoulis, N. and Papadias, D. (2001). Multiway spatial joins. *ACM Transactions on Database Systems (TODS)*, 26(4):424–475.

Mutenda, L. and Kitsuregawa, M. (1999). Parallel r-tree spatial join for a shared-nothing architecture. In *Database Applications in Non-Traditional Environments, 1999.(DANTE'99) Proceedings. 1999 International Symposium on*, pages 423–430. IEEE.

Papadias, D., Mamoulis, N., and Delis, V. (1998). Algorithms for querying by spatial structure. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 546–557. Morgan Kaufmann Publishers Inc.

Papadias, D., Mamoulis, N., and Theodoridis, Y. (1999). Processing and optimization of multiway spatial joins using r-trees. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 44–55. ACM.

Prosser, P. (1993). Hybrid algorithms for the constraint satisfaction problem. *Computational intelligence*, 9(3):268–299.

Zhang, S., Han, J., Liu, Z., Wang, K., and Feng, S. (2009). Spatial queries evaluation with mapreduce. In *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on*, pages 287–292. IEEE.

Projeto e análise de um sistema de nano caches residenciais para distribuição de vídeo*

Gabriel Mendonça[1], **Rosa M. M. Leão**[1]

[1]COPPE/Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro

{gabriel,rosam}@land.ufrj.br

Abstract. *In this paper, we present a P2P video-on-demand (VoD) system that utilizes set-top boxes (STBs) and home gateways for content distribution; we propose a distributed load balancing algorithm that considers the resource constraints in these devices; and describe a probabilistic model that allows a significant reduction of the system's state space: from exponential to polynomial on the number of peers. We evaluate our platform's performance through simulations, using data from a real VoD system in which content popularity seems to follow a Zipf distribution with exponential cutoff. The results show that the proposed algorithm significantly reduces video download time in swarms with low and medium popularity.*

Resumo. *Neste artigo, apresentamos um sistema P2P para transmissão de vídeo sob demanda (VoD) usando* set-top boxes *(STBs) e roteadores domésticos, propomos um algoritmo distribuído para balanceamento de carga que considera a limitação de recursos desses dispositivos e descrevemos um modelo probabilístico que permite a redução do espaço de estados: de exponencial para polinomial no número de* peers*. Avaliamos o desempenho da plataforma através de simulações com dados de um sistema real de VoD, em que a popularidade dos conteúdos é bem modelada por uma distribuição Zipf com decaimento exponencial. Os resultados mostram que, para os* swarms *de baixa e média popularidade, nosso algoritmo reduziu consideravelmente o tempo médio de* download*.*

1. Introdução

As aplicações multimídia têm se tornado cada vez mais populares nos últimos anos. Esse cenário tem motivado diversas propostas para transmissão de vídeo usando arquiteturas *peer-to-peer* (P2P), em que os *peers* participam do processo de distribuição fornecendo recursos para o sistema como banda e capacidade de armazenamento. Aplicações de sucesso como PPLive, QQLive e PPS.tv (PPStream) [1] utilizam essa metodologia para distribuir grandes volumes de tráfego de vídeo na Internet.

Provedores de serviço de Internet (ISP) também podem se beneficiar usando redes P2P. Encontramos na literatura diversas propostas que utilizam os recursos disponíveis em *set-top boxes* (STBs) e/ou roteadores domésticos para a descentralização da distribuição de conteúdo em redes de banda larga [Laoutaris et al. 2008, Cha et al. 2008,

*Este trabalho foi parcialmente financiado pelo CNPq, CAPES e FAPERJ.

[1]http://www.pplive.com/en/index.html, http://live.qq.com/ e http://www.pps.tv/

Whiteaker et al. 2012]. Esse é um cenário favorável para arquiteturas P2P, já que esses dispositivos encontram-se sob controle do provedor de serviço e, em geral, estão ligados durante a maior parte do tempo.

Entretanto, muitos trabalhos [Wu and Lui 2012, Tan and Massoulié 2013] se restringem ao estudo do problema de alocação de conteúdos nos STBs. E, com frequência, as limitações impostas por um dispositivo embarcado não são modeladas com o devido detalhe. Em geral [Suh et al. 2007, Munoz Gea et al. 2012], considera-se apenas um limite no número de conexões de *upload* e *download* que um *peer* é capaz de estabelecer. Além disso, não temos conhecimento de trabalhos sobre redes P2P formadas por STBs que assumiram que a popularidade dos contéudos segue uma distribuição Zipf com decaimento exponencial, como observado em sistemas reais como YouTube e Netflix [Cha et al. 2007].

O objetivo deste trabalho é propor e analisar uma plataforma P2P para transmissão de vídeo sob demanda (VoD) usando nano caches. Os nano caches são dispositivos sob controle de um provedor de serviço de Internet (ISP) presentes na casa dos usuários, como set-top boxes (STBs) e roteadores domésticos. Esses dispositivos formam uma rede P2P, utilizando a banda dos clientes para distribuição de vídeos armazenados localmente. Para o projeto e avaliação dessa plataforma, levamos em consideração circunstâncias reais como uma distribuição de popularidade com decaimento exponencial e uma limitação do número de conteúdos que um dispositivo é capaz de servir simultaneamente.

Nossas principais contribuições incluem:

- Proposta e avaliação de um algoritmo distribuído para balanceamento de carga através da escolha dos conteúdos servidos por cada nano cache.
- Modelo probabilístico para o processo de escolha dos conteúdos servidos, que permite redução significativa do espaço de estados na representação do sistema.
- Análise do desempenho do sistema através de simulações usando parâmetros extraídos de um sistema real de vídeo sob demanda.

Inicialmente, discutimos os trabalhos relacionados na seção 2. Na seção 3, descrevemos a arquitetura do nosso sistema, nosso algoritmo para escolha dos conteúdos a serem servidos e nosso protótipo embarcado desenvolvido para prova de conceito. Em seguida, na seção 4, apresentamos modelos para o limite de capacidade do sistema e para a política de troca de *swarms*. Por fim, apresentamos na seção 5 os resultados de nossas simulações e na seção 6 nossas conclusões.

2. Trabalhos relacionados

Suh *et al.* [Suh et al. 2007] propõem um sistema *peer-to-peer* de VoD usando STBs baseado em operações *push*, em que o conteúdo é "empurrado" pelo provedor de conteúdo para os *peers* em uma etapa anterior à transmissão dos vídeos. Essa proposta, portanto, depende de uma gerência centralizada dos conteúdos armazenados pelos *peers*. Em nosso trabalho, assumimos um algoritmo descentralizado baseado em uma política LRU de substituição de cache, em que um *peer* armazena localmente os conteúdos por ele assistidos.

Em [Munoz Gea et al. 2012], os autores propõem um modelo analítico em que cada STB é modelado como uma fila. Nesse trabalho, o desempenho do sistema é avaliado

a partir da probabilidade de um cliente que assiste a um conteúdo (*leecher*) não encontrar nenhum STB com capacidade disponível para serví-lo. Assim como em [Suh et al. 2007], o número de conexões de *upload* que um STB estabelece é limitado, mas assume-se que um *peer* é capaz de servir qualquer número de conteúdos dentre os que estão armazenados localmente. Em nosso trabalho, assumimos que, devido às limitações de memória e processamento de um dispositivo embarcado, este não é capaz de servir, ao mesmo tempo, todos os conteúdos que armazena.

O problema de colaboração entre *swarms* na distribuição de arquivos com BitTorrent foi estudado em [Chen and Yan 2009, Meng et al. 2013]. Já em [Wang et al. 2010, He et al. 2012, Huang et al. 2008], o mesmo problema foi estudado no contexto de aplicações de VoD P2P. Assim como em nossa proposta, esses autores propõem que a escolha dos conteúdos que um *peer* serve seja feita com base em uma métrica de carga de um *swarm*. A métrica que definimos neste trabalho possui como vantagem o fato de ser facilmente calculada usando informações de um *tracker* ou DHT. As estratégias propostas em [Wang et al. 2010, He et al. 2012], por exemplo, dependem do conhecimento da banda de *upload* dedicada pelos servidores de conteúdo a cada *swarm*, uma informação que não pode ser obtida tão facilmente.

Quanto à popularidade dos conteúdos, grande parte dos trabalhos [Munoz Gea et al. 2012, Wu and Lui 2012, He et al. 2012, Tan and Massoulié 2013, Zhou et al. 2012] assume uma distribuição Zipf. Poucos [Liu et al. 2013] fornecem modelos independentes de distribuição ou usam dados de sistemas reais [Cha et al. 2008, Huang et al. 2008]. Não temos conhecimento de outros trabalhos além do nosso com uma comparação entre o desempenho de um sistema P2P de VoD com uma distribuição Zipf pura e uma distribuição Zipf com decaimento exponencial.

Outros trabalhos [Laoutaris et al. 2008, Whiteaker et al. 2012] sugerem o uso de STBs e *gateways* domésticos num contexto mais geral de aplicações, como VPNs, servidores *web* e jogos *online*.

3. Arquitetura do sistema

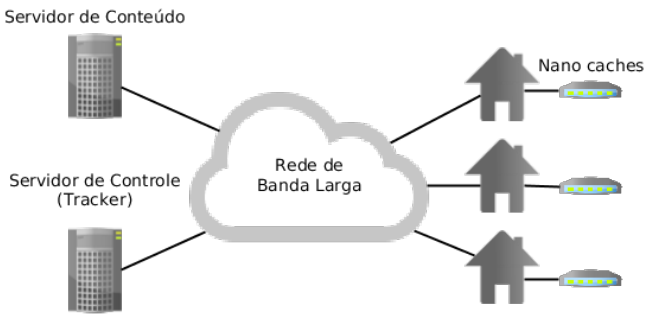


Figura 1. Arquitetura do sistema.

Nesta seção, descrevemos a arquitetura e o funcionamento de nosso sistema, assim como nossa política de troca de *swarms*. Apresentamos também detalhes de nosso protótipo embarcado desenvolvido para prova de conceito.

Propomos um sistema formado por um conjunto de nano caches, um servidor de controle e um servidor de conteúdo, como ilustrado pela Figura 1. Os nano caches são dispositivos embarcados (STBs ou roteadores domésticos) com capacidade de armazenamento local suficiente para um conjunto pequeno de vídeos. Assumimos que estes dispositivos encontram-se em uma mesma rede de banda larga sob controle de um provedor de serviço de Internet (ISP). O servidor de controle é responsável pela indexação dos conteúdos disponíveis, pela autenticação dos clientes e pela monitoração dos nano caches conectados (*tracker*). Já o servidor de conteúdo tem como função garantir a disponibilidade dos vídeos distribuídos no sistema, armazenando cópias de todos os conteúdos e, se necessário, participando da transmissão de vídeo. Mais servidores de conteúdo podem ser adicionados ao sistema para balanceamento de carga e redundância.

Quando um usuário assiste a um vídeo, seu nano cache faz o *download* a partir de outros nano caches (*peers*) e, se preciso, do servidor de contéudo[2]. Em vez de fazer o *download* a uma taxa igual à taxa de codificação do vídeo, o *peer* que recebe o conteúdo (*leecher*) tenta sempre saturar sua banda, minimizando o tempo de *download*. O conteúdo assistido é armazenado localmente e, se não há espaço disponível, uma política LRU (*Least Recently Used*) é adotada. A escolha de uma estratégia ótima para replicação de conteúdos está além do escopo deste trabalho. Já foi mostrado que a adoção de uma política de substituição LRU leva a uma replicação aproximadamente proporcional à popularidade dos conteúdos, fornecendo um desempenho próximo ao ótimo [Zhou et al. 2012].

Os nano caches utilizam a banda de *upload* dos clientes para servir conteúdos que já foram baixados. Como o canal é compartilhado com outros dispositivos da rede doméstica, o nano cache deve minimizar sua interferência com o tráfego local. Se ele atua como roteador doméstico, sua taxa de *upload* pode ser adaptada de acordo com o tráfego medido em suas interfaces de entrada. Caso não seja possível para o nano cache medir diretamente o tráfego local (por exemplo, se for um STB), ele pode utilizar um algoritmo de controle de congestionamento baseado em latência como o LEDBAT [Shalunov et al. 2012]. Ainda assim, esperamos que haja bastante capacidade disponível para o sistema, já que usuários de banda larga raramente saturam sua banda de *upload* e *download* [Grover et al. 2013]. Apresentamos na Tabela 1 a notação utilizada.

3.1. Troca de *swarms*

Dos C conteúdos disponíveis no sistema, um nano cache é capaz de armazenar localmente até B conteúdos, sendo $B \ll C$. Entretanto, devido às suas limitações de processamento e memória, um nano cache serve apenas m dentre os B vídeos armazenados. Precisamos portanto definir uma política para escolha desses m conteúdos que devem ser servidos de modo a fazer uma boa alocação dos recursos disponíveis.

A cada vídeo distribuído no sistema corresponde um *swarm*, do qual participam os *peers* que assistem ao vídeo (*leechers*) e os que o servem (*seeds*). A escolha dos m *swarms* que um *peer* deve servir leva em consideração uma métrica de carga, definida para o *swarm* i como

$$\ell_i = \frac{L_i}{S_i + 1}, \tag{1}$$

[2]A partir deste ponto, adotaremos a terminologia usada em sistemas P2P. Os termos nano cache e *peer* serão utilizados indistintamente.

Tabela 1. Descrição dos principais parâmetros do sistema

Parâmetro	Definição
N	Número de nano caches (*peers*) no sistema
C	Número de conteúdos disponíveis no sistema
B	Número de conteúdos que cabem em um cache
m	Número de conteúdos que um nano cache serve
λ	Taxa de chegada de *leechers*
τ	Intervalo de troca de *swarms*
D_{nc}	Banda disponível para *download* nos nano caches
U_{nc}	Banda disponível para *upload* nos nano caches
U_s	Banda disponível para *upload* no servidor de conteúdo
L_i	Número de *leechers* no *swarm* i
S_i	Número de *seeds* no *swarm* i
p_i	Popularidade relativa do conteúdo i ($\sum_{i=1}^{C} p_i = 1$)
d_i	Duração do vídeo i
b_i	Taxa de codificação (*bitrate*) do vídeo i

onde L_i e S_i são, respectivamente, o número de *leechers* e *seeds* no *swarm* i. Um valor alto de carga pode indicar um número grande de clientes assistindo ao vídeo e/ou um número pequeno de nano caches servindo-o. A carga de um *swarm* pode ser facilmente calculada por um nano cache a partir dos dados recebidos do servidor de controle, que atua como *tracker*. Mesmo sem a presença de um servidor *tracker*, essa métrica poderia ser estimada de maneira descentralizada utilizando, por exemplo, uma Tabela hash distribuída (DHT).

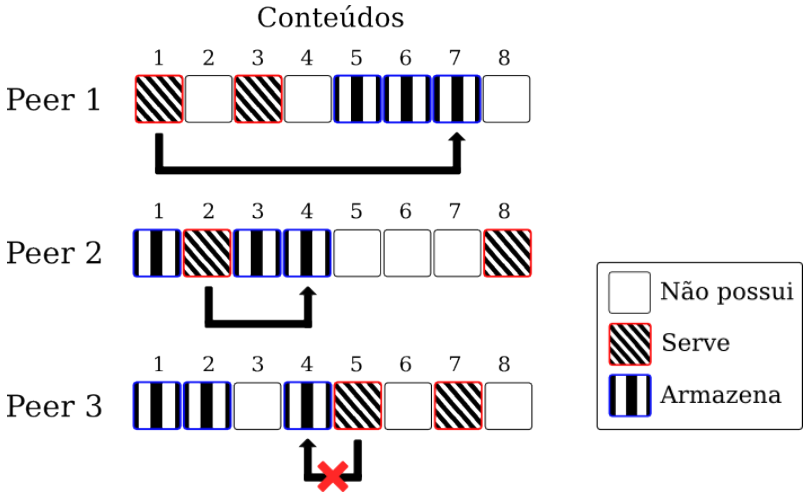


Figura 2. Ilustração da política de troca de *swarms*. Os conteúdos estão ordenados em ordem crescente de carga ℓ_i. Um *peer* deve trocar o *swarm* de menor carga dentre aqueles que serve pelo de maior carga dentre os que armazena (*Peer* 1 troca *swarm* 1 pelo 7 e *Peer* 2 troca *swarm* 2 pelo 4). Caso a carga do primeiro seja maior ou igual à do segundo, a troca não ocorre (*Peer* 3).

A qualidade de experiência do usuário que assiste a um vídeo está relacionada à carga do *swarm* no qual ele é distribuído. Por isso, propomos uma política simples que reduz as diferenças de carga entre os *swarms* introduzindo um *overhead* muito pequeno. Nossa proposta consiste na troca de dois *swarms* em intervalos de tempo τ. No momento da troca, o *peer* busca dentre os m conteúdos que serve aquele que possui o *swarm* de menor carga (*swarm* i). Depois, dentre os $(B - m)$ conteúdos que armazena sem servir,

escolhe aquele com *swarm* de maior carga (*swarm* j). Se $\ell_j > \ell_i$, o *peer* deixa de ser *seed* em i e passa a ser *seed* em j. Caso contrário, ele continua a servir os mesmos m conteúdos. O processo é ilustrado pela Figura 2.

3.2. Protótipo

Desenvolvemos um protótipo de nano cache usando a *libbtstream* [Mendonça and Leão 2012], biblioteca que permite a distribuição de vídeo usando como base o protocolo BitTorrent. O *software* foi adaptado e compilado para o OpenWrt [3], uma distribuição Linux para dispositivos embarcados.

O cliente BitTorrent foi conFigurado para fazer o *download* de pedaços em ordem (sem utilizar a política *Rarest-first* padrão) e fazer *upload* apenas de *torrents* finalizados (como *seed*).

A *libbtstream* faz uso do protocolo uTP [Norberg 2009] de controle de congestionamento no nível da aplicação, que implementa o LEDBAT e minimiza a interferência do tráfego BitTorrent no tráfego doméstico. Isso permite que nosso protótipo de nano cache seja utilizado como roteador *wireless* sem impactar a experiência dos usuários.

Em nossa prova de conceito, usamos como plataforma o roteador doméstico TL-WDR3600 da TP-Link. Esse roteador possui um *System-on-Chip* (SoC) AR9344 da Qualcomm Atheros com processador MIPS operando a 560 MHz, 128 MB de memória RAM, 8 MB de memória flash para armazenamento interno e 2 portas USB 2.0. Para armazenamento dos vídeos, usamos um HD externo com capacidade de 500 GB. Atualmente, possuímos uma rede com nano caches distribuídos entre 15 voluntários de um mesmo provedor de banda larga. Resultados preliminares de experimentos encontram-se em [Mendonça 2015].

4. Modelo

4.1. Limite de capacidade do sistema

Nós propomos um modelo simples que nos permite estimar a taxa máxima de chegada de *leechers* suportada pelo sistema. Com esse modelo, podemos dimensionar os recursos do sistema, como a banda de *upload* dos nano caches e do servidor de conteúdo. Dada uma taxa de chegada, o modelo também nos possibilita o cálculo da taxa de codificação limite para os vídeos, parâmetro de grande impacto na qualidade percebida pelos usuários.

Primeiro, calculamos o número máximo de *leechers* que o sistema pode atender de modo eficiente. Assumiremos que todos os vídeos possuem mesma duração d e taxa de codificação (*bitrate*) b. Para que os clientes sejam capazes de baixar o vídeo a uma taxa maior ou igual a b, queremos que, no pior caso, o valor esperado do número de *leechers* no sistema seja

$$E[L] = \frac{N \cdot U_{nc} + U_s}{b}, \tag{2}$$

onde $N \cdot U_{nc} + U_s$ é a soma da capacidade de *upload* dos nano caches e do servidor de conteúdo. Simultaneamente, queremos que o valor esperado do tempo de *download*, T_D,

[3] https://openwrt.org/

seja no pior caso igual a d. Segundo a Lei de Little,

$$E[L] = \lambda \cdot E[T_D] \iff \lambda = \frac{E[L]}{E[T_D]}. \tag{3}$$

Aplicando (2) em (3) e substituindo $E[T_D]$ pela duração do vídeo, temos nosso limite para a taxa de chegada:

$$\lambda_{max} = \frac{N \cdot U_{nc} + U_s}{b \cdot d}. \tag{4}$$

4.2. Troca de *swarms*

Segundo nossa política de troca de *swarms*, um *peer* deve sair de um dos *swarms* em que é *seed* para entrar em outro *swarm* de carga mais alta. A princípio, para modelar esse processo seria necessário conhecer todos os vídeos que cada nano cache armazena e serve. Entretanto, propomos uma aproximação que nos permite reduzir drasticamente o espaço de estados de um modelo utilizando apenas dois vetores de variáveis de estado de tamanho C, que indicam o número de *seeds* (S_i) e *leechers* (L_i) em cada *swarm*.

Primeiro, modelaremos o processo de saída de um *peer* de um *swarm* de carga mais baixa. Assumimos sem perda de generalidade que os C *swarms* estão ordenados de acordo com sua carga, de modo que $\ell_i \leq \ell_j$ (ver equação (1)) para todo $i < j$, $1 \leq i, j \leq C$. Definimos então a variável aleatória R, que indica o *swarm* de menor carga que um nano cache escolhido ao acaso está servindo. $\Pr(R = k)$ é, portanto, a probabilidade de um *peer* aleatório optar por sair do *swarm* k. Essa probabilidade equivale à fração de nano caches que escolheriam esse *swarm*.

Para que k seja escolhido, é preciso que um *peer* seja *seed* em k e não seja *seed* nos *swarms* anteriores a k (com carga menor ou igual a ℓ_k). Logo, se $\Pr(\omega_i)$ é a probabilidade de um *peer* estar servindo o conteúdo i e $\Pr(\bar{\omega}_i) = 1 - \Pr(\omega_i)$,

$$\Pr(R = k) = \Pr(\omega_k \cap \bar{\omega}_{k-1} \cap \bar{\omega}_{k-2} \cap \cdots \cap \bar{\omega}_1). \tag{5}$$

Propomos uma aproximação para a equação (5) com base na hipótese de que os eventos são mutuamente independentes:

$$\Pr(\hat{R} = k) = c_1 \cdot \Pr(\omega_k) \cdot \prod_{i=1}^{k-1} \Pr(\bar{\omega}_i), \tag{6}$$

onde $c_1 = \left\{ \sum_{j=1}^{C} \Pr(\omega_j) \cdot \prod_{i=1}^{j-1} \Pr(\bar{\omega}_i) \right\}^{-1}$ é uma constante de normalização.

A probabilidade de um nano cache ser *seed* em k é dada por

$$\Pr(\omega_k) = \frac{S_k}{N}, \tag{7}$$

já que S_k dentre os N *peers* do sistema estão servindo o vídeo k.

Aplicando (7) em (6), temos

$$\Pr(\hat{R} = k) = c_1 \cdot \frac{S_k}{N} \cdot \prod_{i=1}^{k-1} \left[1 - \frac{S_i}{N} \right]. \tag{8}$$

Agora iremos modelar o processo de entrada de um *peer* em um *swarm*. Definimos a variável aleatória A, que indica o *swarm* no qual um *peer* aleatório decide entrar. Deste modo, $\Pr(A = k)$ é igual à probabilidade de um nano cache escolhido ao acaso optar por servir o conteúdo k no momento da troca. Vamos assumir que um *peer* escolhe k se e somente se possui k armazenado e não possui os conteúdos posteriores a k (de carga maior ou igual a ℓ_k) [4]. Se $\Pr(\sigma_i)$ é a probabilidade de um nano cache ter o conteúdo i e $\Pr(\bar{\sigma}_i) = 1 - \Pr(\sigma_i)$, temos que

$$\Pr(A = k) = \Pr(\sigma_k \cap \bar{\sigma}_{k+1} \cap \bar{\sigma}_{k+2} \cap \cdots \cap \bar{\sigma}_C) \tag{9}$$

Assumimos novamente que os eventos são mutuamente independentes. Deste modo, temos a aproximação

$$\Pr(\hat{A} = k) = c_2 \cdot \Pr(\sigma_k) \cdot \prod_{i=k+1}^{C} \Pr(\bar{\sigma}_i) \tag{10}$$

onde $c_2 = \left\{ \sum_{j=1}^{C} \Pr(\sigma_j) \cdot \prod_{i=j+1}^{C} \Pr(\bar{\sigma}_i) \right\}^{-1}$ é uma constante de normalização.

Aproximamos a probabilidade de um nano cache possuir o conteúdo k por uma distribuição binomial. Para isso, assumimos que os B conteúdos em um nano cache são amostrados de maneira independente (com reposição) e que o conteúdo k é escolhido com probabilidade p_k:

$$\Pr(\sigma_k) \approx 1 - (1 - p_k)^B. \tag{11}$$

Aplicando (11) em (10), temos:

$$\Pr(\hat{A} = k) = c_2 \cdot \left[1 - (1 - p_k)^B\right] \cdot \prod_{i=k+1}^{C} (1 - p_i)^B. \tag{12}$$

Tendo escolhido os *swarms* r e a para sair e entrar, respectivamente, o *peer* efetua a troca de *swarms* se e somente se $\ell_d < \ell_a$. Caso contrário, ele permanece como *seed* nos mesmos *swarms* por pelo menos mais τ unidades de tempo.

As aproximações $\hat{R}$ e $\hat{A}$ (equações (8) e (12)) foram avaliadas através do método Monte Carlo. Obtivemos um *Root Mean Squared Error* (RMSE) normalizado de 1,38% $\pm$ 0,02 p.p. para $\hat{R}$ e 2,77% $\pm$ 0,05 p.p. para $\hat{A}$.

Com essas duas variáveis aleatórias, podemos modelar a decisão tomada por um *peer* sem conhecer o seu estado, ou seja, os m *swarms* que ele serve e os B conteúdos que possui. Se representamos o estado de cada *peer*, o espaço de estados cresce exponencialmente com o número de nano caches. Com nossa aproximação, o crescimento é polinomial [Mendonça 2015]. Portanto, a aproximação é essencial para resolver modelos com números de *leechers* e *seeds* iguais aos de sistemas reais.

[4] Não consideramos aqui os conteúdos que o *peer* está servindo no momento. Se, por exemplo, já estivesse servindo k, este não poderia ser escolhido.

5. Avaliação de desempenho

Nesta seção, apresentamos uma análise do desempenho de nosso sistema com o objetivo de: (i) validar o modelo para o limite de capacidade do sistema; (ii) observar o impacto da política de troca de *swarms*; (iii) avaliar o funcionamento com e sem a participação do servidor de conteúdo; e (iv) medir a influência de diferentes distribuições de popularidade.

Para tal, desenvolvemos um modelo usando a ferramenta Tangram-II [de Souza e Silva et al. 2009], que possui como variáveis de estado o número de *leechers* e *seeds* em cada *swarm* e usa nossas aproximações $\hat{R}$ e $\hat{A}$ para representar o processo de troca de *swarms*. Assumimos que a taxa de chegada de *leechers* no sistema segue um processo de Poisson homogêneo e que a probabilidade de um novo *leecher* escolher o conteúdo k é igual a p_k. Além disso, assumimos que todos os *peers* saturam sua banda de *upload*, que é dividida igualmente entre os m conteúdos que servem, e que todos os vídeos possuem a mesma duração (5400 s) e taxa de codificação (5 Mbps). Esta análise pode ser facilmente estendida para o caso mais geral em que há diferença de duração e *bitrate* entre os conteúdos. E, a fim de simplificar nossa análise, modelamos o tempo entre trocas de *swarms* e o tempo de *download* dos conteúdos como variáveis aleatórias exponenciais. Essa é uma das principais fontes de aproximação do modelo. Os valores adotados por padrão para os parâmetros do modelo podem ser vistos na Tabela 2.

A distribuição de popularidade dos conteúdos foi extraída de dados de um sistema de VoD de um grande provedor brasileiro de banda larga. Obtivemos acesso a *logs* que cobrem 38299 usuários durante um período de 3 meses. Testes de razão de verossimilhança [Vuong 1989] indicaram, com p-valor próximo a zero, que uma distribuição Zipf com decaimento exponencial é mais adequada à distribuição empírica do que uma Zipf pura, que segue uma lei de potência e é usada com muita frequência para modelar popularidade de conteúdos na Internet (ver seção 2). O decaimento exponencial da cauda da distribuição de popularidade foi observado em outros sistemas reais, como o Netflix, o Youtube e o chinês PowerInfo [Cheng et al. 2007, Cha et al. 2007]. A PMF de uma distribuição Zipf com decaimento exponencial é dada por $\Pr(Z = k) = c \cdot k^{-\alpha} \cdot e^{-\mu \cdot k}$, onde c é uma constante de normalização. Utilizando MLE, estimamos os parâmetros $\hat{\alpha} = 0,63$ e $\hat{\mu} = 0,02$.

Em nossa análise, usamos as seguintes métricas de desempenho: $\bar{D}$, a taxa média de *download* dos *peers*; $\bar{T}_D$, o tempo médio de *download*; e $\bar{G}$, a fração média de tempo com taxa de *download* maior do que a taxa de codificação. Todos os resultados são apresentados com um intervalo de confiança de 95% para 20 rodadas de simulação.

Tabela 2. Parâmetros padrão adotados nas simulações

Parâmetro	Valor	Parâmetro	Valor
N	40000	D_{nc}	10 Mbps
C	250	U_{nc}	1 Mbps
B	20 ($\approx$ 66 GB de armazenamento)	U_s	1 Gbps
m	4	p_i	$\propto i^{-0,63} \cdot e^{-0,02 \cdot i}$
λ	4800 *leechers* / h	d_i	5400 s (90 min) para todo i
τ	1800 s (30 min)	b_i	5 Mbps para todo i

5.1. Taxa de chegada

A fim de validar nosso modelo para o limite de capacidade e avaliar a escalabilidade do sistema, realizamos simulações mantendo a taxa de *upload* do servidor de conteúdo igual

a 1 Gbps e variando o valor de λ. Nesse cenário, temos um limite para a taxa de chegada de aproximadamente 5467 *leechers* por hora. Os resultados são apresentados na Tabela 3.

Vemos que o sistema apresenta um excelente desempenho para valores abaixo do limite superior de escalabilidade. Mesmo para uma taxa de chegada próxima ao limite, de 5200 *leechers* por hora, observamos, por exemplo, que em média 99% dos usuários experimentam tempo de *download* inferior a 5400 segundos. Esses resultados sugerem uma boa qualidade de experiência para os usuários, com pouco ou nenhum tempo de *buffering*.

Para taxas de chegada acima do limite, notamos que a qualidade se degrada rapidamente, já que os nano caches e o servidor de conteúdo não possuem banda suficiente para atender à demanda. Isso se reflete na taxa média de *upload* menor do que 5 Mbps, no alto tempo de *download* e na probabilidade próxima a zero de um *leecher* possuir uma taxa de *download* maior do que a taxa de codificação por mais de 95% do tempo.

Tabela 3. Avaliação do valor máximo de λ. Limite superior: 5467 *leechers* / h

λ (*leechers* / h)	$\bar{D}$ (Mbps)	$\Pr(\bar{T}_D \leq 5400s)$	$\Pr(\bar{G} > 95\%)$
4800	$9,91 \pm 0,04$	$0,9988 \pm 2,78 \cdot 10^{-4}$	$0,97 \pm 6,83 \cdot 10^{-4}$
5200	$8,64 \pm 0,26$	$0,9949 \pm 1,14 \cdot 10^{-3}$	$0,98 \pm 6,79 \cdot 10^{-4}$
5600	$3,73 \pm 0,15$	$0,0022 \pm 1,31 \cdot 10^{-4}$	$0,02 \pm 0,04$
6000	$2,60 \pm 0,05$	$0,0013 \pm 1,11 \cdot 10^{-4}$	$0,00 \pm 0,00$

5.2. Troca de *swarms*

Mostramos agora os resultados de nossa avaliação da política de troca de *swarms*. Comparamos nossas métricas de desempenho para diferentes valores de τ. No cenário em que não há uma política de troca de *swarms*, quando um *leecher* termina o *download* de um conteúdo ele passa a serví-lo imediatamente e deixa de servir um dos m conteúdos que está servindo. Nesse caso, o *swarm* a ser deixado é escolhido aleatoriamente de acordo com a probabilidade do *peer* ser *seed* naquele *swarm* (ver equação (7)).

Os resultados apresentados na Tabela 4 sugerem uma piora de todas as métricas de desempenho quando não há troca de *swarms*. Observamos também uma pequena degradação da qualidade com o aumento do valor de τ (tempo entre trocas).

Na Figura 3, vemos um gráfico com o tempo médio de *download* ($\bar{T}_D$) por *swarm*. Notamos uma piora no desempenho quando não há troca de *swarms*. Sem nossa política, os *swarms* de baixa e média popularidade apresentam tempo médio de *download* até 3 vezes maior do que a duração do vídeo. Entretanto, para os 40 conteúdos mais populares, não vemos uma diferença significativa. Vemos também que o valor de $\bar{T}_D$ cresce juntamente com τ nos *swarms* de menor popularidade.

Tabela 4. Avaliação do intervalo entre trocas de *swarms*.

Política	$\bar{D}(Mbps)$	$\Pr(\bar{T}_D \leq 5400s)$	$\Pr(\bar{G} > 95\%)$
Sem troca	$8,31 \pm 0,09$	$0,9372 \pm 5,46 \cdot 10^{-3}$	$0,83 \pm 0,02$
30 min	$9,91 \pm 0,04$	$0,9988 \pm 2,78 \cdot 10^{-4}$	$0,97 \pm 6,83 \cdot 10^{-4}$
120 min	$9,79 \pm 0,04$	$0,9976 \pm 3,43 \cdot 10^{-4}$	$0,97 \pm 6,01 \cdot 10^{-4}$
480 min	$9,45 \pm 0,05$	$0,9933 \pm 5,04 \cdot 10^{-4}$	$0,97 \pm 5,06 \cdot 10^{-3}$

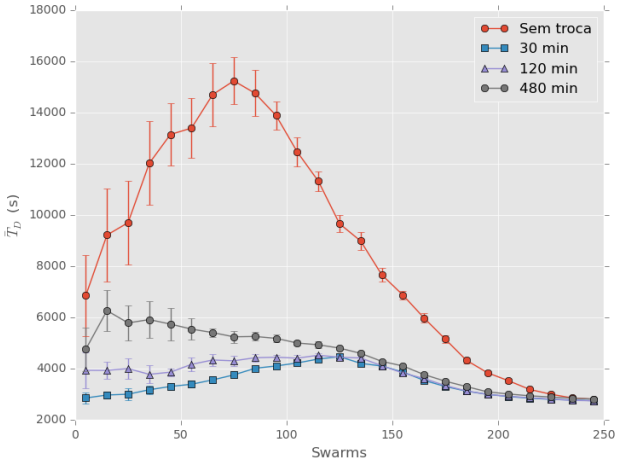


Figura 3. Comparação do tempo médio de *download* por *swarm* para diferentes valores de τ. *Swarms* em ordem crescente de popularidade.

Concluímos que nossa política de troca de *swarms* causa grande impacto na qualidade percebida pelos usuários. Acreditamos que o valor do parâmetro τ pode ser ainda mais relevante em cenários com variação súbita de popularidade (*flash crowd*).

5.3. Servidor de conteúdo

Nos voltamos agora para a análise do impacto da participação do servidor de conteúdo na transmissão dos arquivos de vídeo. Para isso, executamos simulações com U_s igual a 0 Gbps. Na comparação com os resultados para U_s igual a 1 Gbps, vemos uma diferença pouco significativa na taxa média de *download* dos *leechers*: $9,83 \pm 0,08$ Mbps sem servidor e $9,91 \pm 0,04$ Mbps com servidor.

A comparação do tempo médio de *download* por *swarm* também indica pouco impacto do servidor de conteúdo nesse cenário. Como pode ser visto na Figura 4, o cenário sem servidor apresentou valores de $\bar{T}_D$ maiores para os conteúdos menos populares. Os vídeos para os quais observamos um tempo médio de *download* maior representam cerca de 2% das visualizações do sistema. Ainda assim, o tempo médio de *download* manteve-se menor do que a duração dos vídeos em todos os *swarms*.

5.4. Distribuição de popularidade

Por fim, analisamos o desempenho de nosso sistema em função da distribuição de popularidade dos conteúdos. Comparamos a distribuição Zipf com decaimento exponencial com uma distribuição Zipf pura com parâmetro $\alpha = 0,8$.

Na Figura 5, vemos o tempo médio de *download* nos *swarms* medido para as duas distribuições. Vemos uma grande diferença no formato das curvas. O sistema com uma distribuição Zipf pura apresenta pior desempenho para os *swarms* menos populares. Já no caso da distribuição com decaimento exponencial, os usuários tendem a experimentar um tempo de *download* maior ao assistirem aos conteúdos de popularidade média. Lembramos que, com uma distribuição Zipf com truncamento exponencial, os 100 vídeos menos populares somam cerca de 1% dos acessos ao sistema. Com a distribuição Zipf, os 100 conteúdos de menor popularidade recebem aproximadamente 14% das visualizações.

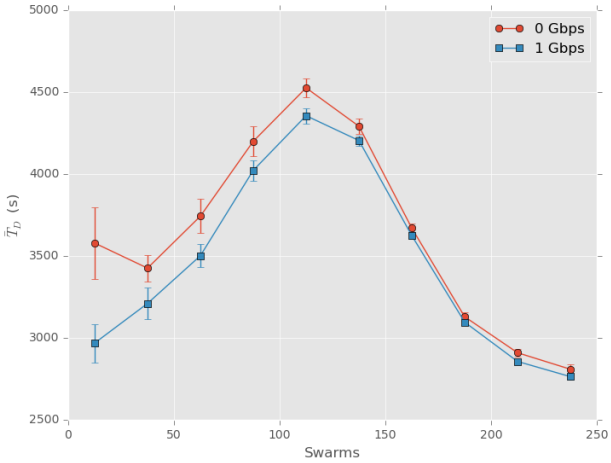


Figura 4. Comparação do tempo médio de *download* por *swarm* com e sem a participação do servidor de conteúdo. *Swarms* em ordem crescente de popularidade.

Concluímos que a aproximação da distribuição real de popularidade do sistema por uma distribuição Zipf pode não ser boa, já que há uma variação significativa no desempenho quando comparada a uma distribuição com decaimento exponencial, mais próxima da realidade.

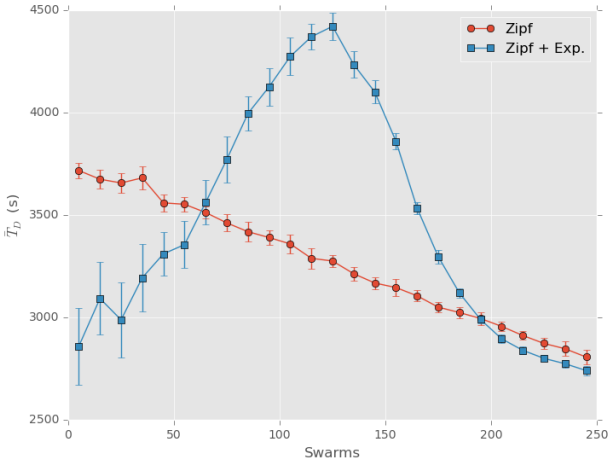


Figura 5. Comparação do tempo médio de *download* por *swarm* para as duas distribuições de popularidade. *Swarms* em ordem crescente de popularidade.

6. Conclusão

Neste trabalho, apresentamos uma plataforma P2P para transmissão de vídeo usando dispositivos embarcados projetada e avaliada com base em circunstâncias reais. Definimos uma política para escolha dos conteúdos a serem servidos por um nano cache que permite o balanceamento da carga do sistema de forma distribuída sem a adição de *overheads*, usando uma métrica simples de carga de um *swarm*.

Através de simulações, avaliamos o desempenho do sistema em diversos cenários e pudemos constatar o impacto de nosso algoritmo de troca de *swarms* na distribuição dos conteúdos menos populares. Ressaltamos que, embora tenhamos assumido nano caches com mesma banda de *upload* e *download* e vídeos de mesma duração e *bitrate*, nosso estudo pode ser facilmente estendido para o caso mais geral.

Nosso modelo probabilístico para a política de troca de *swarms* permite uma redução expressiva no espaço de estados do sistema que, a princípio, deveria crescer exponencialmente com o número de nano caches. Desse modo, torna-se viável a análise do desempenho do sistema usando parâmetros próximos aos reais. No futuro, pretendemos usar esse modelo para a dedução de novos resultados analíticos.

Acreditamos ser este o primeiro trabalho a apresentar uma análise do desempenho de uma plataforma P2P de VoD em que a popularidade de seus conteúdos segue uma distribuição Zipf com decaimento exponencial, com parâmetros extraídos de um sistema real. Concluímos que ainda há muito a ser investigado em cenários como esse. Em especial, observamos que, devido à brusca queda na cauda da distribuição de popularidade, novas medidas podem ser necessárias para atender adequadamente aos usuários interessados em conteúdos de nicho.

Esperamos no futuro poder utilizar nosso protótipo embarcado de nano cache para montar um *testbed* e realizar experimentos com o objetivo de confirmar os resultados de nossas simulações e validar nossas premissas.

Referências

Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., and Moon, S. (2007). I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proc. of ACM IMC*, pages 1–14. ACM.

Cha, M., Rodriguez, P., Moon, S. B., and Crowcroft, J. (2008). On next-generation telco-managed P2P TV architectures. In *Proc. of IPTPS*, page 5.

Chen, Z. and Yan, J. (2009). Optimizing resource scheduling in BitTorrent file distribution network. In *Proc. of MASS'09*, pages 1–4. IEEE.

Cheng, X., Dale, C., and Liu, J. (2007). Understanding the Characteristics of Internet Short Video Sharing: YouTube as a Case Study. *CoRR*, abs/0707.3670.

de Souza e Silva, E., Figueiredo, D. R., and Leão, R. M. (2009). The TANGRAM-II integrated modeling environment for computer systems and networks. *ACM SIGMETRICS PER*, 36(4):64–69.

Grover, S., Park, M. S., Sundaresan, S., Burnett, S., Kim, H., Ravi, B., and Feamster, N. (2013). Peeking behind the NAT: an empirical study of home networks. In *Proc. of ACM IMC*, pages 377–390.

He, Y., Xiong, Z., Zhang, Y., Tan, X., and Li, Z. (2012). Modeling and analysis of multi-channel P2P VoD systems. *Journal of Network and Computer Applications*, 35(5):1568–1578.

Huang, Y., Fu, T. Z., Chiu, D.-M., Lui, J. C., and Huang, h. (2008). Challenges, Design and Analysis of a Large-scale P2P-vod System. *ACM SIGCOMM CCR*, 38(4):375–388.

Laoutaris, N., Rodriguez, P., and Massoulie, L. (2008). ECHOS: edge capacity hosting overlays of nano data centers. *ACM SIGCOMM CCR*, 38(1):51–54.

Liu, F., Li, B., Li, B., and Jin, H. (2013). Peer-assisted on-demand streaming: Characterizing demands and optimizing supplies. *IEEE TC*, 62(2):351–361.

Mendonça, G. and Leão, R. (2012). BTStream – Um ambiente para desenvolvimento e teste de aplicações de streaming P2P. In *Proc. of SBRC*.

Mendonça, G. (2015). Sistema de nano caches residenciais para distribuição de vídeo. Master's thesis, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

Meng, X., Tsang, P.-S., and Lui, K.-S. (2013). Analysis of distribution time of multiple files in a P2P network. *Computer Networks*, 57(15):2900–2915.

Munoz Gea, J. P., Traverso, S., and Leonardi, E. (2012). Modeling and evaluation of multisource streaming strategies in P2P VoD systems. *IEEE T-CE*, 58(4):1202–1210.

Norberg, A. (2009). uTorrent Transport Protocol (BEP 29). `http://www.bittorrent.org/beps/bep_0029.html`. Acessado em: 29/11/2014.

Shalunov, S., Hazel, G., Iyengar, J., and Kuehlewind, M. (2012). Low Extra Delay Background Transport (LEDBAT). RFC 6817 (Experimental).

Suh, K., Diot, C., Kurose, J., Massoulie, L., Neumann, C., Towsley, D., and Varvello, M. (2007). Push-to-peer video-on-demand system: Design and evaluation. *IEEE J-SAC*, 25(9):1706–1716.

Tan, B. and Massoulié, L. (2013). Optimal content placement for peer-to-peer video-on-demand systems. *IEEE TON*, 21(2):566–579.

Vuong, Q. H. (1989). Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, pages 307–333.

Wang, Z., Wu, C., Sun, L., and Yang, S. (2010). Strategies of collaboration in multi-channel P2P VoD streaming. In *Proc. of IEEE GLOBECOM*, pages 1–5. IEEE.

Whiteaker, J., Schneider, F., Teixeira, R., Diot, C., Soule, A., Picconi, F., and May, M. (2012). Expanding home services with advanced gateways. *ACM SIGCOMM CCR*, 42(5):37–43.

Wu, W. and Lui, J. (2012). Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation. *IEEE TPDS*, 23(8):1492–1503.

Zhou, Y., Fu, T. Z., and Chiu, D. M. (2012). A unifying model and analysis of P2P VoD replication and scheduling. In *Proc. of IEEE INFOCOM*, pages 1530–1538. IEEE.

Consolidation of VMs to improve energy efficiency in cloud computing environments

Thiago Kenji Okada[1], **Albert De La Fuente Vigliotti**[1], **Daniel Macêdo Batista**[1], **Alfredo Goldman vel Lejbman**[1]

[1]Institute of Mathematics and Statistics (IME) – University of São Paulo (USP)
São Paulo, SP, Brazil

{thiagoko,albert,batista,gold}@ime.usp.br

Abstract. *Improvement of energy efficiency in IT is an important research topic nowadays. The reduction of operational costs, generated heat and environment impact are some of the reasons for this. Thanks to the advent of cloud computing, it is possible to improve energy efficiency in data centers by running various virtual machines in a single physical machine. However, the cloud providers generally invest in performance, not energy efficiency. This paper focuses on the problem of an energy efficient initial VM placement, and describes three new algorithms for this problem, one based on the First Fit Decreasing algorithm, and the other two based on the Best Fit Decreasing algorithm. They are compared with other algorithms in the literature, and a reduction of power consumption up to 3.24% was observed, as well a reduction of execution time in several orders of magnitude. Scripts used to analyze traces publicly provided by Google are another contribution of the paper, since they are useful for those working in mechanisms for cloud computing.*

1. Introduction

The advent of cloud computing is changing the way investments in computing resources are made. The characteristics of cloud computing services, including the illusion of infinite computing resources, elimination of upfront costs, and the ability to pay per use [Armbrust et al., 2010], made those kinds of services popular. Different service providers are rapidly deploying new data centers to keep up with the increasing demand for cloud services.

Performance was the sole concern in cloud environments, to keep up with the demand, while energy consumption did not receive much attention [Beloglazov et al., 2012]. However, worldwide energy usage in data centers accounted for between 1.1% and 1.5% of total electricity use in 2010 [Koomey, 2011]. As the energy costs keep increasing, focus on improving the energy efficiency in data centers is important, but it is necessary to maintain similar performance while doing so.

Different approaches to improve energy efficiency may be applied on data centers, including development of applications that use resources more efficiently, and use of modern hardware, including technologies like Dynamic Voltage and Frequency Scaling (DVFS) [Beloglazov and Buyya, 2010]. Nevertheless, an efficient approach is to reduce the amount of hardware over-provisioning by the implementation of more efficient VM (Virtual Machine) placement algorithms.

Virtualization technology allows several VMs in a single physical machine. However hardware is typically over-provisioned in order to sustain peaks of resource demand during short-periods of time, since cloud providers are bound to Service-Level Agreements (SLAs) [Feller et al., 2011]. The result of this strategy is an average low resource utilization of data centers: for example, in [Barroso and Hölzle, 2007] is shown that the mean value of CPU utilization in data centers is 36.44% with 95% confidence interval.

Therefore, improving VM placement can lead to interesting gains in energy efficiency. For example, if we can allocate more VMs using less resources, the tendency is to reduce the power consumption of the data center.

This paper focuses on an energy efficient initial VM placement, instead of optimizing an existing VM placement like other works in the literature. This work presents three different algorithms, one based on the First Fit Decreasing (FFD) algorithm [Yue, 1991] and two based on the Best Fit Decreasing (BFD) algorithm [Suoerv and Johnson, 1973] to solve the problem of VM placement, including modifications that allow each algorithm to suspend physical hosts and that evaluate the impact of a new VM allocation in the entire data center energy consumption. We modeled the VM placement problem as a bin packing problem [Beloglazov and Buyya, 2012; Vigliotti and Batista, 2014b].

We implemented and evaluated these algorithms in pyCloudSim[1], a simulation framework written in the Python language by the authors, and compared them with other algorithms in the literature [Beloglazov and Buyya, 2012; Vigliotti and Batista, 2014b] using traces available from Google's clusters [Reiss et al., 2011]. Before presenting the results of the experiments, an extensive analysis of the Google's traces is detailed. The scripts developed to make this analysis represent another contribution of this paper and they are publicly available. In the experiments, our algorithms performed better than other algorithms presented in the literature, while being multiple times faster.

The remainder of this paper is structured as follows: In Section 2, we review the literature about the area. In Section 3, we describe the problem and our proposed algorithms. In Section 4, we describe the simulation environment, including the analysis of Google's traces. In Section 5, we present the performance evaluation. Finally, in Section 6, we present the conclusion of this work and future directions.

2. Related work

[Barroso and Hölzle, 2007] describe techniques to reduce the power consumption in servers. These techniques are directly related with our simulation model, that uses dynamic-voltage-frequency scaling (DVFS) and sleeping states to reduce power usage. DVFS reduces voltage according to the current CPU frequency, reducing overall consumption, while sleeping states put the host in a state of low power consumption and disable host functionalities until "wake up" by changing its state.

[Beloglazov and Buyya, 2012; Beloglazov, 2013] present the Power Aware Best Fit Decreasing (PABFD) which is based on the Best Fit Decreasing algorithm. However, the focus of those studies is in the optimization of an existing VM placement, instead of focusing in the initial VM placement problem. Therefore it does not overlap with our

[1] `https://github.com/vonpupp/pyCloudSim/tree/v0.1` . Accessed on December 14, 2014

study. Other differences between the PABFD and our algorithms based on BFD is that our algorithms allows physical hosts to suspend and, also, because one of our algorithms evaluate the impact of a new VM allocation in terms of the entire data center energy consumption.

[dos Santos Júnior, 2014] shows an implementation of FFD for the VM placement problem. While it does a comprehensive study of the problem, it does not compare FFD with other algorithms, as we do in this work. The difference between the implementation proposed in [dos Santos Júnior, 2014] and our algorithm based on FFD is that our algorithm allows physical hosts to suspend.

[Vigliotti and Batista, 2014a] present two approaches for the problem of VM placement. The first one (called *Iterated-KSP*) is an iterated approach for a Knapsack based heuristic, solving the problem n-times (once per host). The second one (called *Iterated-EC*) is based on an Evolutionary Computation model, using an iterated approach. Experiments in Section 5 show that our algorithms are faster and perform better than the algorithms presented in [Vigliotti and Batista, 2014a].in [Vigliotti and Batista, 2014a] by including new algorithms in the comparison and adding support to Google's traces data.

In [Vigliotti and Batista, 2014b] the authors present two variations to the algorithms presented in [Vigliotti and Batista, 2014a]. The first one, called (*Iterated-KSP-Mem*), is a modification of Iterated-KSP that doubles the size of virtual memory to allocate VMs that consume too much virtual memory. The second one, called (*Iterated-EC-Net*), is a modification of Iterated-EC aiming to reduce both the power consumption and network utilization, instead of only the former. While we did not include Iterated-EC-Net in our comparison, since Google's traces data does not include network usage information, our experiments show that our algorithms are faster and perform better than the other algorithm presented in [Vigliotti and Batista, 2014b].

To the best of our knowledge, there is no work in the literature that compares different algorithms for the initial VM placement problem and there is no algorithm to this objective that runs faster and more efficiently than the algorithms we are proposing in this paper.

3. Energy efficient VM placement

The energy-efficient VM placement problem can be divided in two sub-problems. The first one is the placement of a requested VM on a host, while the second one is the optimization of the current VM placement. The focus on this work is the first one.

Similar to [Feller et al., 2011; Vigliotti and Batista, 2014a], we modeled our problem as a instance of the well-known multi-dimensional bin-packing (MDBP) problem. To find the optimal solution of a MDPB problem is NP-hard [Jung et al., 2010], therefore it is interesting to use heuristics instead of calculating the exact solution, for efficiency reasons, since a slow algorithm would not be interesting in real data centers receiving several requests to allocate new VMs at every second.

An energy efficient VM placement aims to reduce the number of active physical machines, by increasing the workload in the current active machines. The idea is simple: if the machine already has work to do, increasing the amount of work will not increase the power consumption of the data center significantly. However, waking up a new machine

will impact the power consumption of the data center more significantly, since a machine in suspended state consumes a little fraction of the power of an active one. Even in the case when the data center is composed by several modern machines that do not have a consumption directly proportional to the usage, our work is valid, since we consider how much Watts are consumed by each one of the operation modes of the processor(s) of the machine.

The strategy used to place the VMs must be careful to not overload a host, since it is important to not violate any Service-Level Agreements (SLAs) between the cloud provider and the client.

The next subsections present the three proposed algorithms. The first algorithm (Subsection 3.1) and the second algorithm (Subsection 3.2) modify existing algorithms by including instructions related to the suspension of physical machines. The third algorithm (Subsection 3.3) extends existing algorithms by including instructions related to suspension of physical machines and also including instructions to evaluate if a new VM allocation affects the entire data center energy consumption.

3.1. The First Fit Decreasing algorithm

The first algorithm presented in this work is a modification of the greedy algorithm described in the literature [Yue, 1991; Esnault, 2012], called First Fit Decreasing. The pseudo-code for the algorithm is shown in Algorithm 1.

```
   Input: hostList, vmList
   Output: A VM placement
1  Sort vmList in the order of decreasing utilization;
2  foreach vm in vmList do
3      foreach host in hostList do
4          if host has enough resources for vm then
5              Allocate vm in host;
6              break;
7  foreach host in hostList do
8      if host has no allocated vm then
9          Suspend host
```

Algorithm 1: The First Fit Decreasing (FFD) algorithm for VM placement.

This algorithm sorts the VMs on the list of requests into non-increasing order by CPU usage (line 1) and then processes the list in that order by placing each VM into the first host which fits (lines 2-5). It continues to the next VM (line 6) until all VMs are allocated. Finally, lines 7-9 suspend idle physical hosts, to reduce power consumption.

[Yue, 1991] shows that the result given by this algorithm for a list L of items is no more than $\frac{11}{9}OPT(L) + 1$, where $OPT(L)$ is the optimal solution. Our modification of FFD is power aware: if no VMs are allocated to a host, it suspends that host, greatly reducing the energy consumption.

3.2. The Power Aware Best Fit Decreasing algorithm

The second algorithm is based on the work from Beloglazov, called Modified Best Fit Decreasing (MBFD) [Beloglazov and Buyya, 2012] or, more recently, Power Aware Best Fit Decreasing (PABFD) [Beloglazov, 2013]. According to [Beloglazov and Buyya, 2012], this algorithm is a variation of FFD algorithm, and the same demonstration from [Yue, 1991] is still valid.

The algorithm is shown in Algorithm 2. While the algorithm itself is similar to FFD, the main difference is that PABFD calculates the increase in power consumption after the allocation of the current VM (line 12) in each host in data center (lines 7-14), storing the lowest increase in power consumption (lines 15-17). The VM is then allocated to the host that had the lowest increase in power consumption (lines 18-19). Finally, lines 20-22 actually suspend idle physical hosts, similar to Algorithm 1.

We modified the PABFD algorithm from [Beloglazov, 2013] to consider that every host is suspended before the initial allocation (lines 1-2). This change made the allocation of a VM in a unused host very expensive in terms of energy consumption, since it would need to wake-up the host first (lines 9-11), and allows our algorithm to prioritize hosts that already had VMs allocated to them.

3.3. The Global Power Aware Best Fit Decreasing algorithm

Our third algorithm is a modification of the algorithm described in Section 3.2. In this modification, we changed the function that returns the estimate in power consumption after a VM allocation, i.e., we changed the function $estimateIncreaseInPower$ in line 13 from Algorithm 2 to the code showed in Algorithm 3.

Instead of looking to the power consumption increase after the allocation of a new VM, our algorithm calculates the power consumption of the whole data center (lines 3-4) after simulating an allocation of VM in the current host (line 2). We named it *Global Power Aware Best Fit Decreasing* (GPABFD) due to this.

4. The simulation framework and workloads

We evaluate the performance of our proposed algorithms through simulation. We noticed that the available cloud computing simulators like CloudSim [Buyya et al., 2013] make their decision related to VM placement only based on CPU usage. Because of that, we extended the pyCloudSim simulator [2]. The simulator is a free software over the Apache 2.0 license.

Although the simulator was previously used to evaluate the performance of VM placement mechanisms in [Vigliotti and Batista, 2014a,b], on those works it used traces from PlanetLab machines, which not necessarily represent the higher diversity in workloads encountered in cloud computing. This fact motivated us to extend the pyCloudSim by including support to the Google cluster traces, traces that captures different workloads (Subsection 4.1).

Presenting the details of pyCloudSim is out of the scope of this paper, nevertheless more details about the framework can be found in [Vigliotti and Batista, 2014a]. We will

[2] `https://github.com/m45t3r/pyCloudSim`. Accessed on December 14, 2014.

```
Input: hostList, vmList
Output: A VM placement
1  foreach host in hostList do
2      Assume host is suspended;
3  Sort vmList in the order of decreasing utilization;
4  foreach vm in vmList do
5      minPower ← ∞;
6      allocatedHost ← Null;
7      foreach host in hostList do
8          if host has enough resources for vm then
9              if host is suspended then
10                 hostWasSuspended ← True;
11                 Assume Wake-on-Lan on host;
12             power ← estimateIncreaseInPower(host, vm);
13             if hostWasSuspended = True then
14                 Assume host is suspended;
15         if power < minPower then
16             allocatedHost ← host;
17             minPower ← power;
18     if allocatedHost ≠ Null then
19         Allocate vm in host;
20 foreach host in hostList do
21     if host has no allocated vm then
22         Suspend host
```

Algorithm 2: The Power Aware Best Fit Decreasing (PABFD) algorithm for VM placement.

```
Input: host, vm
Output: Power consumption of data center
1  totalPower ← 0;
2  Simulate allocation of vm in host;
3  foreach host in hostList do
4      totalPower ← totalPower + estimatePower(host);
5  Remove vm from host;
```

Algorithm 3: The Global Power Aware Best Fit Decreasing (GPABFD) modification.

focus, in Subsection 4.1, to provide information about the Google's traces used in our experiments, which is useful to researchers working with cloud computing mechanisms and needing more information about these traces to execute experiments.

4.1. Google cluster traces

Cloud computing has higher diversity in workloads profiles than traditional data centers. One big, public cloud provider may have different kinds of applications running in its cloud, including big data analysis, Web services, scientific computing, software deployment and automated tests, and others.

However, several traces of cluster data report more homogeneous workload. For example, one trace may have a great number of high-performance, non-latency sensitive tasks (like big data analysis), while other is more focused in long-running servers like Web servers. Google made available traces from one of its computing clusters that captures different behavior, including the former examples and others [Reiss et al., 2012].

Google cluster traces data is composed by various *jobs*. Each job is created by a *user*, and it is comprised of one or more *tasks*, including the requirements used for scheduling the tasks on physical machines. There is no known distribution that fits these data, but the resources appear to form a heavy-tailed distribution [Reiss et al., 2011].

The raw information of the current version of Google cluster data is composed of 40GB of compressed data. Some tasks have 0 units of CPU, RAM, Disk I/O or Disk space [Reiss et al., 2011] and some tasks are really long, including tasks running for the whole period of the traces. So we needed to filter the data before using it. To do this, we created a script to import the raw data available in the Google website to a SQL database[3], allowing us to query for various pieces of data included in the traces.

We need to make some assumptions on the data to do a proper simulation. Google does not provide details about their computing environment and the data itself is obfuscated, so we do not know if the application is running on a VM, a container or even on the real OS. Each job has a multitude of tasks, that may or not run in the same machine. However, a task is a single Linux program, possibly consisting of multiple processes, to be run on a single machine [Reiss et al., 2011]. So it is reasonable to consider each task as a single VM instance or a container.

Another assumption is about machine resources. Each resource data (including CPU, memory, disk space and I/O time) is normalized, relative to the largest capacity of the resource on any machine in the traces. So we need to assume that our environment is homogeneous (i.e. all machines have the same specifications), even when Google's cluster environment is heterogeneous [Reiss et al., 2011].

According to the official website of the traces [Wilkes and Reiss, 2013], the current version of the traces data represent 29 days of data from May 2011 on a cluster of 11k machines. The trace files are divided in 500 parts, from 00000 to 00499. Disk time data is only included in the first 14 days, because of a change in the monitoring system. So we evaluated the first 50 parts, from 00000 to 00049, that still included disk I/O time data. This is equivalent to 2713386 unique tasks. About 1% of the jobs in the traces

[3] `https://gist.github.com/m45t3r/0242be48a82bafee23f1` . Accessed on December 14, 2014.

have missing resource records, so we excluded them. We did not considered the network resource on this work, due to the fact that the Google Cluster Data traces do not include this resource metric.

From the rest of the traces data, we filtered the entries that had an average CPU usage from 15% to 80%, to get only tasks that had relevant CPU usage. This is equivalent to 3725 unique tasks. We then selected the first 288 tasks to represent the VMs for our simulation, since initial experiments showed that this was sufficient to show the difference between the algorithms presented in this work. This sample is sufficient since [Reiss et al., 2012] showed that there is no known distribution that seems to good fit the Google traces data. The possible explanation is due to unpredictability of humans factors in resource usage.

The histogram of CPU usage for these selected tasks is presented in Figure 1. The majority of the tasks, roughly 75% of selected tasks or about 2800 tasks, are in the 15%-20% range of average CPU usage. The rest of tasks, or approximately 25% of selected tasks, falls in the range starting from 20% of average CPU usage up to 45%, characterizing a heavy-tail distribution.

The histogram of memory usage for these selected tasks is in Figure 2. The majority of the tasks, roughly 67% of selected tasks or about 2500 tasks, are in the 0%-10% range of average memory usage. Roughly 29% of selected tasks or about 1100 tasks, are in the 10%-20% range. About 4% of the tasks falls in the range starting from 20% of average memory usage up to 60%, again characterizing a heavy-tail distribution.

The histogram of disk I/O time for these selected tasks is in Figure 3. It is interesting to note that disk usage is almost irrelevant in Google's traces. The majority of tasks, roughly 67% of selected tasks or about 2500 tasks, are in the 0%-0.5% range of average disk I/O time. About 33% of the tasks falls in the range starting from 0.5% of average disk I/O time up to 3%, characterizing another heavy-tail distribution.

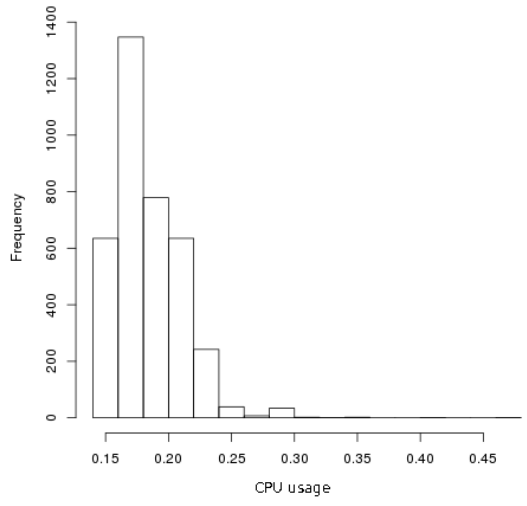


Figure 1. Histogram of Number of tasks X Average CPU Rate

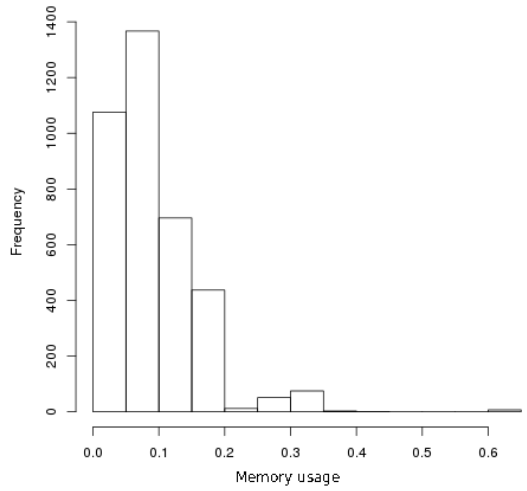


Figure 2. Histogram of Number of tasks X Average Memory Usage

5. Performance evaluation

The hardware used on the experiments was a virtual machine hosted in USP Cloud[4] scientific infrastructure. The physical machine has an Intel® Xeon® CPU E7- 2870 @ 2.40GHz. The virtual machine has 8 virtual CPU cores, 32GB of allocated RAM and two virtual disks, one of 20GB and another one of 500GB. The code and data of all experiments is available on GitHub[5].

We compared the proposed algorithms (FFD, PABFD and GPABFD) with an algorithm called Energy-Unaware, which simulates a First Come First Served (FCFS) algorithm without energy awareness. In this algorithm, we firstly randomize the VM request list to simulate requests coming from users. Another comparison is done with all algorithms available in [Vigliotti and Batista, 2014b], excluding Iterated-EC-Net, due to the fact that the Google traces do not include network usage information.

We have two different simulation scenarios, one with 35 physical machines and another one with 100 physical machines, each one starting with 16 VMs up to 288, in 16 VMs increments. The first scenario simulates a heavily used cloud, using more resources than are available. The second scenario simulates a more common situation, where the cloud provider overprovision their resources, but generally have more than sufficient resources for all the current tasks running in the cloud.

Each physical machine used the power model described in [Vigliotti and Batista, 2014a], that uses a linear power-to-frequency relationship for a server. For the real power usage we use data provided from SPECpower benchmark[6]. Each experiment was repeated 30 times to calculate the average and confidence intervals with a level of confidence equal

[4] `http://www.sti.usp.br/?q=node/5370` . Accessed on December 14, 2014.

[5] `https://github.com/m45t3r/sbrc-2015-simulation` . Accessed on December 14, 2014.

[6] `http://www.spec.org/power_ssj2008/results/res2014q3/power_ssj2008-20140615-00658.html` . Accessed on December 14, 2014.

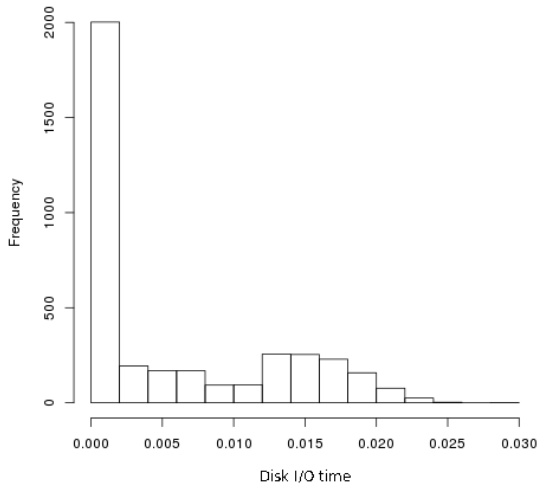


Figure 3. Histogram of Number of tasks X Average Disk I/O Time

to 95%.

Figure 4 shows the average number of VMs placed in our first simulation scenario, with 35 hosts. The FFD, PABFD and GPABFD algorithms performs worse than the rest of the algorithms in this situation where there is not sufficient resources for all VMs. With 288 VMs, for example, FFD, PABFD and GPABFD allocates 25.5% less VMs than Iterated-KSP, the best algorithm in this experiment. This performance of FFD, PABFD and GPABFD can be explained since these algorithms have a greedy nature, and once a decision is made they do not optimize a solution further.

Figure 5 shows the average number of VMs placed with 100 hosts, and confirms that there is sufficient resources for all VMs, at least for the initial allocation. In this case, all the algorithms performed similarly.

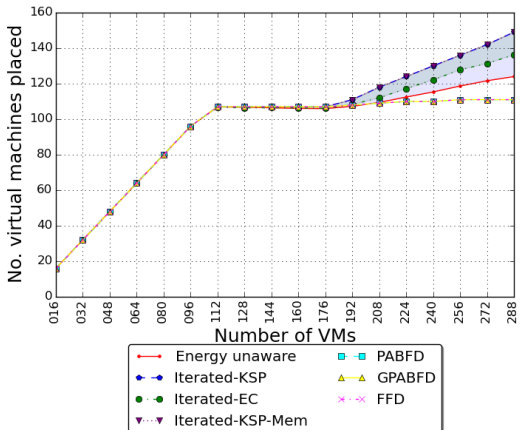


Figure 4. Average placed VMs comparison (35 hosts)

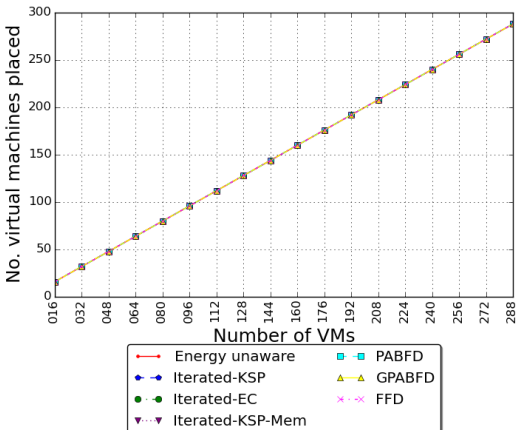


Figure 5. Average placed VMs comparison (100 hosts)

Figure 6 shows the average power consumption with 35 hosts. All algorithms

excluding Energy-Unaware have similar consumption. This result is confirmed by Figure 8, where all algorithms excluding Energy-Unaware suspended a similar number of VMs.

Figure 7 shows the average power consumption with 100 hosts. FFD, PABFD and GPABFD performs similarly, but performs better than Iterated-KSP, Iterated-KSP-Mem, Iterated-EC and Energy-Unaware after 192 VMs. FFD for example, consumed 3.24% less power than Iterated-KSP in the 288 VMs case. Even including the confidence intervals (at most $\pm 0.53\%$ with 95% confidence), FFD, PABFD and GPABFD consumed less power than the other algorithms. This result is confirmed by Figure 9, where our algorithms suspended more VMs than the Iterated-KSP, Iterated-KSP-Mem, Iterated-EC and Energy-Unaware, starting from 192 VMs.

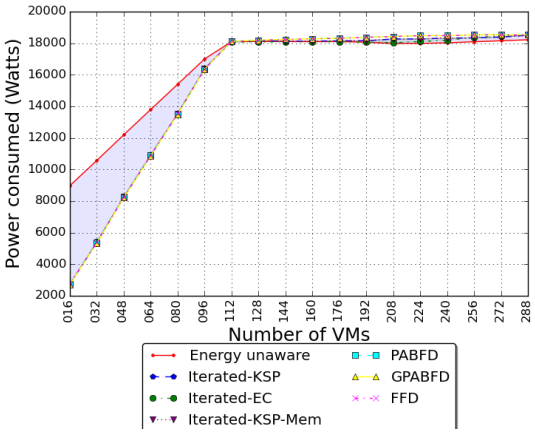


Figure 6. Average energy consumption comparison (35 hosts)

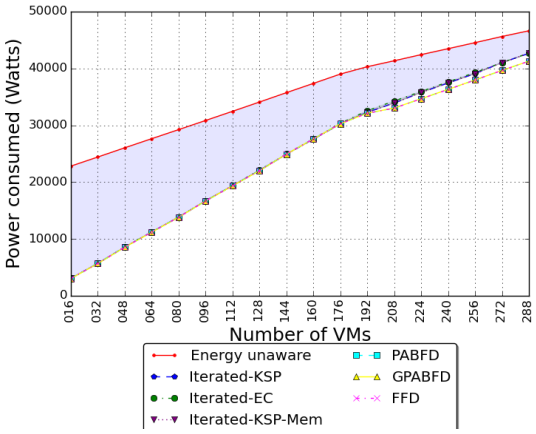


Figure 7. Average energy consumption comparison (100 hosts)

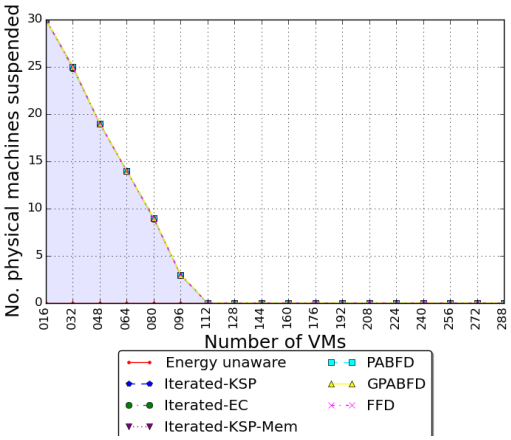


Figure 8. Average number of physical machines suspended comparison (35 hosts)

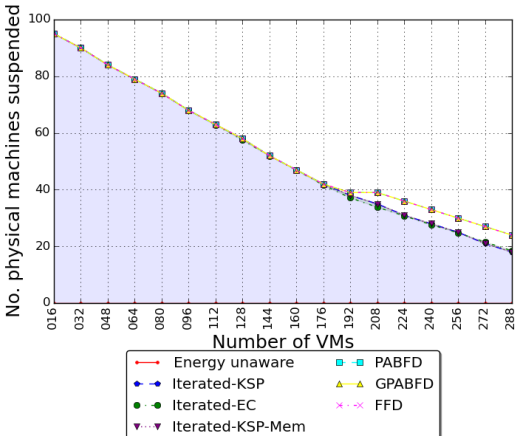


Figure 9. Average number of physical machines suspended comparison (100 hosts)

Figures 10 and 11 show the execution time of each algorithm. Since the FFD, PABFD and GPABFD algorithms are simpler than other algorithms, they are faster even with a great number of VMs. For example, FFD uses less than 2% of the time used by Iterated-KSP or Iterated-EC in the case with 288 VMs and 100 hosts. Even including the confidence intervals (at most $\pm 52.98\%$ with 95% confidence), FFD, PABFD and

GPABFD are at least one order of magnitude faster than the other algorithms. The FFD algorithm in particular shows execution times below one second in general. The high range of confidence intervals for execution time may be explained by performance variation between measurements, since the simulation was running in a cloud environment.

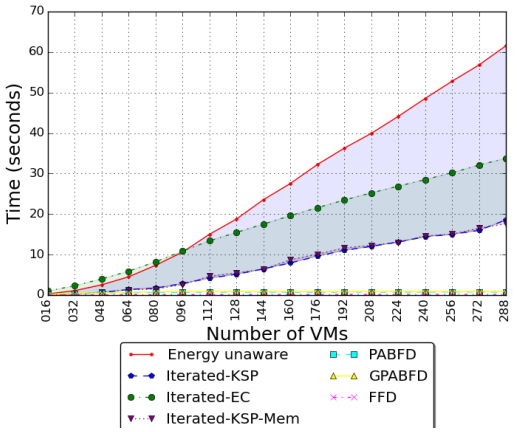


Figure 10. Average execution time comparison (35 hosts)

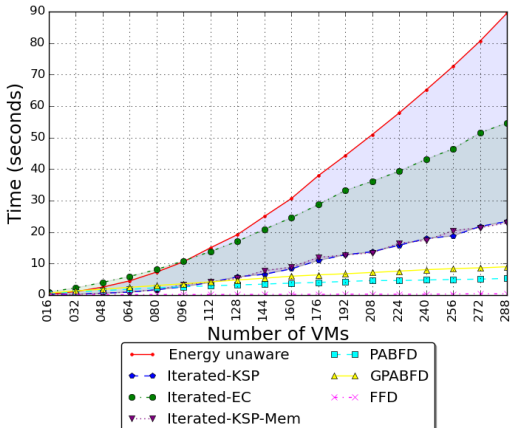


Figure 11. Average execution time comparison (100 hosts)

Table 1 summarizes the average execution time for all algorithms in the most demanding scenario simulated, with 100 hosts and 288 VMs. As explained, the FFD, PABFD and GPABFD algorithms are faster than all other algorithms by at least one order of magnitude.

Table 1. Average execution time with 100 hosts and 288 VMs

Strategy	Average time
Energy-Unaware	89.48
Iterated-KSP	23.38
Iterated-KSP-Mem	22.90
Iterated-EC	54.64
FFD	0.38
PABFD	5.30
GPABFD	8.99

6. Conclusion and future directions

The Fit-Decreasing family of algorithms (FFD, GABFD and PABFD) have an interesting performance according to the experiments. In a cloud with heavy usage, where there is not sufficient resources for every virtual machine currently in the system, they allocate less VMs than all other algorithms, including Energy-Unaware. However, it can be argued that this situation is not realistic, since it would mean that the cloud provider did not have sufficient resources to serve all its clients.

In a more realistic situation, where there is sufficient resources for every virtual machine, the Fit-Decreasing family of algorithms performs even better than more sophisticated algorithms like Iterated-KSP and Iterated-EC from [Vigliotti and Batista, 2014a].

They are very fast too, with execution times some orders of magnitude lower than the other existing algorithms.

Comparing the three proposed algorithms (FFD, PABFD and GPABFD), there is not much difference in energy consumption, but FFD is still faster than PABFD and GPABFD. This works concludes that FFD algorithm is an interesting candidate for cloud providers being both efficient in real workloads and fast.

For future work, there are other candidates to solve the problem of VM placement, including ant colonies optimization, dynamic programming and sharing aware algorithms. These algorithms could be compared with FFD, PABFD and GPABFD to show their advantages and disadvantages compared to our approach.

Another proposal for future work is to implement support in pyCloudSim to optimize an existing VM allocation, instead of just the initial allocation. This will probably need support for multithreading capabilities, so each physical machine run in a separate thread instead of the current solution of only one thread.

References

Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., and Rabkin, A. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50.

Barroso, L. A. and Hölzle, U. (2007). The Case for Energy-Proportional Computing. *IEEE Computer Society*, (December):33–37.

Beloglazov, A. (2013). *Energy-efficient management of virtual machines in data centers for cloud computing*. PhD thesis, The University of Melbourne.

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.

Beloglazov, A. and Buyya, R. (2010). Energy Efficient Resource Management in Virtualized Cloud Data Centers. *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 826–831.

Beloglazov, A. and Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420.

Buyya, R., Calheiros, R. N., and Nikolay, G. (2013). CloudSim website. `www.cloudbus.org/cloudsim/`. Accessed: 2013-12-04.

dos Santos Júnior, M. R. (2014). Mecanismos para consolidação de servidores. Master's thesis, Universidade de São Paulo.

Esnault, A. (2012). Energy-Aware Distributed Ant Colony Based Virtual Machine Consolidation in IaaS Clouds.

Feller, E., Rilling, L., and Morin, C. (2011). Energy-aware ant colony based workload placement in clouds. In *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID '11, pages 26–33, Washington, DC, USA. IEEE Computer Society.

Jung, G., Hiltunen, M., Joshi, K., Schlichting, R., and Pu, C. (2010). Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In *Proceedings of the 30th IEEE ICDCS*, pages 62–73.

Koomey, J. G. (2011). Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*.

Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7.

Reiss, C., Wilkes, J., and Hellerstein, J. L. (2011). Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, pages 1–14.

Suoerv, T. and Johnson, D. S. (1973). *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology.

Vigliotti, A. P. M. d. l. F. and Batista, D. M. (2014a). Energy-efficient virtual machines placement. In *Proceedings of the SBRC 2014*, pages 1–8.

Vigliotti, A. P. M. d. l. F. and Batista, D. M. (2014b). A green Network-Aware VMs placement mechanism. In *Proceedings of the IEEE Globecom 2014*, page To appear.

Wilkes, J. and Reiss, C. (2013). Google Cluster Data Website. `https://code.google.com/p/googleclusterdata/wiki/ClusterData2011_2`. Accessed: 2014-12-12.

Yue, M. (1991). A simple proof of the inequality $FFD\,(L) \leq 11/9OPT\,(L)+1, \forall\, L$ for the FFD bin-packing algorithm. *Acta mathematicae applicatae sinica*, 7(4):321–331.

Uma Proposta de Controle de Congestionamento Ocasionado pela Comunicação Máquina-a-Máquina em LTE

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. *M2M communications is a communication model used by devices where data can be exchanged with little or no human intervention. The M2M, when applied in the context of LTE network, can lead to both overload and congestion problems due to its intrinsic particularities. Accordingly, in this paper we propose a new congestion control approach for LTE that reduces the impact on the H2H devices and establishes priority amongst M2M devices through the use of priority classes. The distinction amongst these classes is established based on the access probability and the backoff interval. Besides, the distinction amongst devices of the same class is defined based on the number of access attempts by the device. The results obtained through extensive simulations in NS-3 show that the proposed approach can control the impact on H2H, establishes priorities intra and interclass of M2M and reduces access delay. Besides, it has high feasibility of implementation.*

Resumo. *O modelo de comunicação M2M é utilizado por dispositivos para troca de informações com pouca ou nenhuma intervenção humana. O M2M, quando aplicado no contexto da rede LTE, pode ocasionar problemas de sobrecarga e congestionamento devido as suas particularidades. Diante desse cenário, o presente artigo tem como objetivo propor uma abordagem de controle de congestionamento para o LTE que reduz o impacto sobre os dispositivos H2H e estabelece priorização entre dispositivos M2M com a criação de classes. A distinção entre as classes apresentadas neste trabalho é estabelecida segundo a probabilidade de acesso e o intervalo de backoff. Além disso a distinção entre os dispositivos de uma mesma classe é definida a partir do número de tentativas de acesso realizadas pelo dispositivo. Os resultados obtidos através de exaustivas simulações no NS-3 mostram que a proposta controla o impacto sobre H2H, estabelece priorização intra e interclasse do M2M, reduz o tempo de acesso e apresenta alta viabilidade de implementação.*

1. Introdução

A comunicação Máquina-a-Máquina (*Machine-to-Machine* - M2M) deve possibilitar a troca de informações fim-a-fim entre um ou mais dispositivos autônomos com pouca ou nenhuma intervenção humana [Maia et al. 2014]. Dentre esses dispositivos, estão equipamentos de propósito comum (e.g., eletrodomésticos, automóveis, celulares, etc.) e específicos (e.g., sensores, atuadores, etc.). O uso massivo da comunicação M2M estimado para os próximos anos, servindo de base para Internet das Coisas (*Internet of Things* –

IoT), possibilitará o surgimento de aplicações que facilitarão atividades comuns do dia a dia, gerando um profundo impacto na sociedade [Chang et al. 2014]. Nesse contexto, a infraestrutura LTE (*Long Term Evolution*) se apresenta como potencial candidata para implantação da comunicação M2M.

A rede LTE apresenta vantagens como mobilidade, acessibilidade, cobertura, segurança, dentre outras consideradas relevantes para implantação de serviços e aplicações M2M [Hasan et al. 2013]. Por serem redes inicialmente projetadas para a comunicação Humano-a-Humano (*Human-to-Human* – H2H) e Humano-a-Máquina (*Human-to-Machine* – H2M), alguns aspectos da rede LTE precisam ser adaptados para suportar o M2M. Ciente desse problema, a *3rd Generation Partnership Project* (3GPP) tem apresentado requisitos (e.g. controlar o impacto sobre o tráfego H2H) considerados críticos para viabilizar o M2M na rede LTE [3GPP 2012]. Um dos problemas identificados pela 3GPP é a sobrecarga e o congestionamento, discutido na Seção 2, que ocorrem na rede de acesso por rádio (*Radio Access Network* – RAN) do LTE durante o procedimento de acesso aleatório ao canal (ou procedimento de RACH (*Random Access CHannel*)) ocasionado pelo excesso de dispositivos M2M.

A sobrecarga e o congestionamento na RAN são problemas que ocorrem no LTE devido à limitação de *slots* de acesso (*RA-Slots*) e de códigos de preâmbulo disponíveis. O emprego de um procedimento de acesso aleatório ao canal apresenta problemas de sobrecarga e congestionamento, representando assim um dos gargalos na rede LTE para implantação do M2M [Zheng et al. 2014]. Com o objetivo de guiar as propostas de controle do congestionamento na RAN, a 3GPP propõe seis abordagens: *Access Class Barring* - ACB, *Backoff*, *Separate* RACH, *Dynamic Allocation*, *Slotted-Access*, *Pull* e *Backoff* [3GPP 2014a]. Algumas abordagens na literatura (e.g. [Lo et al. 2011], [Lin et al. 2014], [Lee et al. 2011]) combinam duas ou mais dessas técnicas na tentativa de equilibrar os aspectos negativos existentes em cada técnica. A Seção 3 apresenta os trabalhos relacionados.

Contudo, poucas soluções consideram o tráfego H2H no tratamento da sobrecarga [Zheng et al. 2014]. A falta de compatibilidade com o padrão 3GPP é também outro aspecto que distancia tais propostas da arquitetura da rede LTE, dificultando a integração e consequente implantação em um contexto prático [Jian et al. 2013]. Esses problemas motivaram o desenvolvimento do mecanismo proposto neste artigo. Assim sendo, o presente trabalho, discutido na Seção 4, apresenta um mecanismo para controle de congestionamento de fácil implementação que reduz o impacto sobre o H2H. Além disso, a abordagem proposta apresenta distinção entre dispositivos M2M de alta e baixa prioridade. Para isso, uma abordagem que altera a probabilidade de acesso e tempo de *backoff* é utilizada para as classes de dispositivos. Os resultados obtidos através de exaustivas simulações utilizando o simulador NS-3, que são apresentados e discutidos na Seção 5.2, mostram que a proposta apresenta bons resultados na priorização entre os dispositivos e baixa complexidade de implantação. Finalmente, na Seção 6 apresenta-se as conclusões e possíveis trabalhos futuros.

2. Fundamentação Teórica

2.1. Comunicação Máquina-a-Máquina

A comunicação M2M possui aplicabilidade nas mais diversificadas áreas. Algumas características normalmente encontradas nas aplicações e serviços M2M são [Maia et al. 2014]: (i) tráfego principal no sentido ascendente, (ii) transmissão esporádica e (iii) transmissão de dados de pequeno porte. Assim, para atender a essas e outras características (e.g., baixa mobilidade, tempo controlado, tolerância ao atraso, agrupamento entre os dispositivos, etc.) normalmente apresentadas nas aplicações e serviços M2M, a rede LTE precisa ser aprimorada para o modelo de comunicação M2M.

2.2. Procedimento de Acesso ao Canal

Com a introdução dos dispositivos M2M em LTE mais dispositivos irão competir por recursos para acessar a rede acarretando aumento no número de colisões. As colisões durante o procedimento de RACH ocorrem quando dois ou mais dispositivos escolhem o mesmo código de preâmbulo para o envio de solicitação de acesso à rede. Em caso de colisão, o dispositivo atinge o máximo de espera e entra em estado de *backoff*. Diante deste cenário, faz-se necessário apresentar as etapas que compõem o procedimento de acesso aleatório do LTE.

Na rede LTE, dois métodos de acesso são permitidos: um livre de contenção e um baseado em contenção. No primeiro, o acesso é iniciado pela estação base (*evolved-NodeB* – eNB). No segundo, o acesso é iniciado pelo dispositivo[1] que solicita acesso à rede quando [Laya et al. 2014]: (i) precisa se registrar na rede, (ii) solicita recursos no canal de *uplink*[2] para transferência de dados, (iii) durante o processo de *handover* e (iv) após uma falha no link de rádio, para restabelecimento de conexão.

O acesso ao canal no LTE (ou procedimento de RACH) é composto de quatro mensagens de sinalização (Msg1, Msg2, Msg3 e Msg4, ilustradas na Fig. 1) gerenciadas pelo controlador de recursos de rádio (*Radio Resource Control* - RRC). A Fig. 1 ilustra a troca de mensagens que ocorre entre um dispositivo e uma estação base durante o procedimento de RACH baseado em contenção.

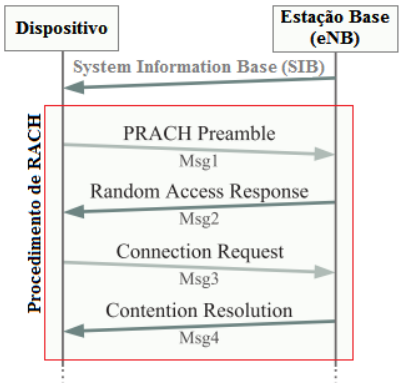


Figura 1. Procedimento de acesso à rede LTE baseado em contenção.

[1]Quando não especificado, refere-se aos dispositivos M2M e H2H.
[2]Canal Compartilhado de Uplink - *Physical Uplink Shared Channel* – PUSCH

Antes de solicitar acesso à rede, o dispositivo identifica como os *slots* de acesso (*RA-slots*) estão mapeados no canal. Para isso, o dispositivo aguarda as mensagens de SIB (*System Information Block*), difundidas periodicamente pela estação base. A partir das configurações recebidas, o dispositivo identifica a paridade do *frame* e os *subframes* nos quais as solicitações de acesso podem ser transmitidas. Na sequência, as seguintes atividades são realizadas em cada etapa do procedimento de acesso aleatório ao canal:

1. PRACH Preamble (Msg1) – Nesta etapa, o dispositivo escolhe aleatoriamente um código de preâmbulo para envio da solicitação.

2. Random Access Response - RAR (Msg2) – Nesta etapa, ocorre: (i) detecção do preâmbulo de acesso aleatório na eNB, (ii) atribuição de um identificador único e temporário (*Temporary Cell Radio Network Temporary Identifier* - TC-RNTI) para o dispositivo e (iii) concessão de um canal de *uplink* para transmissão das mensagens subsequentes.

3. Connection Request (Msg3) – Nesta etapa, o dispositivo envia o identificador temporário que lhe foi atribuído na etapa anterior. Os dados são enviados para a estação base (*evolved NodeB* - eNB) através do canal compartilhado de *uplink* (*Physical Uplink Shared Channel* – PUSCH) seguindo as configurações definidas na etapa anterior.

4. Contention Resolution (Msg4) – Nesta etapa, a eNB envia para o dispositivo uma mensagem de resolução de contenção através do canal compartilhado de downlink (*Downlink Shared Channel* - DL-SCH). O dispositivo aguarda o recebimento da resposta e analisa se nesta consta o seu identificador; em caso afirmativo, o dispositivo responde para a eNB com uma mensagem de confirmação (ACK); caso contrário, aguarda o término do tempo de *backoff* para poder retransmitir.

3. Trabalhos Relacionados

Em [Jian et al. 2013] os autores propõem a priorização de dispositivos variando o tempo de *backoff* e criando classes de *backoff* para cada tipo de dispositivo. Em caso de colisão, o dispositivo aguarda o tempo de *backoff* escolhido aleatoriamente ($X \sim Unif(0, T)$) de acordo com a classe à qual pertence, sendo $Unif(\alpha, \beta)$ uma distribuição uniforme no intervalo (α, β). Os autores apresentam três abordagens para determinar o intervalo das classes. Na primeira, o intervalo (T) é dividido em (C) classes com intervalo definido a partir do nível de congestionamento (p_0) informado pela eNB. Os dispositivos H2H apresentam intervalo de valores de *backoff* fixo e uniformemente distribuído entre 0 e T. Além disso, os dispositivos M2M apresentam intervalo de valores (0, T_{bf}^{MTC}), onde T_{bf}^{MTC} é calculado através da equação: $T_{bf}^{MTC}(i) = p_0 T + Unif(i\frac{(1-p_0)T}{C}, (i+1)\frac{(1-p_0)T}{C})$. Para evitar que o intervalo determinado para as classes M2M convirja para zero com o aumento do nível de congestionamento, os autores propõem alterações no cálculo do tempo de *backoff* calculado através da equação: $T_{bf}^{MTC}(i) = T + Unif(i\frac{kT}{C}, (i+1)\frac{kT}{C})$.

Em [Jian et al. 2013], a criação de classes de *backoff* possibilita a priorização entre os dispositivos, o que representa uma melhora em relação à estratégia de *backoff* convencional e ao *Slotted Aloha*. A implementação é simples, com alterações no

modo como os dispositivos realizam o cálculo de *backoff*. Entretanto, a abordagem em [Jian et al. 2013] apresenta desperdício de recursos com o aumento na diferença do número de dispositivos por classe, pois divide as classes de *backoff* em intervalos de mesmo tamanho, independente do número de dispositivos. Além disso, não estabelece priorização entre dispositivos M2M de uma mesma classe.

Em [Lee et al. 2011] os autores propõem dois modos de divisão dos slots de acesso aleatório entre os dispositivos H2H e M2M, sendo um dinâmico e o outro estático. Antes de transmitir, os dispositivos são selecionados utilizando o *Access Class Barring* - ACB. Além da complexidade de implementação, a abordagem apresentada pelos autores não estabelece nenhum tipo de priorização entre os dispositivos M2M. Hasan *et al* (2013) contribui indiretamente para o controle do congestionamento com um algoritmo de seleção de eNB. O algoritmo emprega técnicas de aprendizagem (observa, aprende e adapta) e considera os requisitos de QoS antes de escolher a eNB. Apesar de apresentar fácil implementação, a proposta não estabelece nenhum tipo de priorização entre os dispositivos M2M. Em [Lo et al. 2011], os autores propõem controlar a sobrecarga com a alocação dinâmica de recursos para o procedimento de RACH. Para isso, propõem um algoritmo que detecta o nível de sobrecarga na RAN. O algoritmo identifica o nível de sobrecarga de acordo com o número de solicitações de acesso realizadas pelo dispositivo até a obtenção de acesso à rede. Nessa proposta, outras técnicas (*Slotted-Access*, *Backoff* e ACB) também são combinadas para minimizar o impacto sobre o H2H e estabelecer priorização entre dispositivos M2M. Além da alta complexidade de implementação, a proposta não foi validada, seja por simulação ou analiticamente. Outras estratégias para simular o congestionamento são apresentadas por Chen *et al* (2013) que propõem um *test-bed framework* para análise de estratégias de controle de congestionamento de M2M em rede LTE.

O congestionamento nas redes LTE devido ao M2M pode ocorrer também em outros pontos da rede como apresentado em [Lam et al. (2013)]. Contudo, as estratégias de controle de congestionamento empregadas na borda da rede (RAN) apresentam-se como soluções mais eficazes [Hasan et al. 2013].

4. Proposta de Controle de Congestionamento

Nesta seção, apresenta-se uma proposta para controle de congestionamento que visa: (i) reduzir o impacto sobre H2H, (ii) estabelecer priorização intra e interclasses entre os dispositivos H2H e M2M e (iii) aumentar a taxa de acesso[3].

4.1. Problema

O procedimento de RACH da rede LTE segue os princípios do *Slotted Aloha* [Laya et al. 2014]. Devido à analogia existente entre o *Slotted Aloha* e o procedimento de RACH do LTE, pode-se dizer que a probabilidade de colisão (P_c) durante o procedimento de RACH na rede LTE é aproximado pela equação [Phuyal et al. 2012, 3GPP 2014a]:

$$P_c = 1 - e^{(\frac{-\lambda}{L})} \quad (1)$$

[3]Número de acessos à rede efetuados com sucesso.

na qual L é o total de *slots* de acesso (*RA-Slots*) por segundo e λ a quantidade média de solicitações realizadas por segundo para uma estação base.

Para um largura de banda (B) dada em MHz, tal que $B \in \{1.4, 2.5, 5, 10, 15, 20\}$ a quantidade de blocos de recursos físicos (*Physical Resource Blocks* - PRBs) disponíveis por *frame* é dada por: $PRB_{TotalFrame} = \frac{B}{F_{subframe}} \times T_{frame} \times 2$, sendo T_{frame} o tempo do *frame* (dado em milissegundo), $F_{subframe}$ a largura de banda (dada em kHz) de um PRB do LTE. Sabendo que os *slots* de acesso (*RA-Slots*) podem ser configurados para ocorrer n vezes em um T_{frame}, com $n \in \{0.5, 1, 2, 3, 5, 10\}$. Assim, tem-se que a quantidade de PRBs disponíveis para o procedimento de RACH por segundo determinado por:

$$PRB_{Total} = \frac{PRB_{TotalFrame}}{2} \times \frac{1000}{T_{frame}} \times n. \tag{2}$$

Assim, o número de solicitações de acesso suportado pela estação base (eNB) por segundo é dado por:

$$T_{RAR} = \frac{PRB_{Total}}{PRB_{RACH}} \tag{3}$$

onde PRB_{RACH} é a quantidade de blocos de recursos físicos ($PRBs$) utilizados por solicitação de RACH, sendo igual a 6 nas redes LTE.

Para uma taxa de colisão (P_c), o número de *slots* de acessos necessários para suportar a intensidade de acesso é dado em [3GPP 2014a] por:

$$\lambda = -L \times ln(1 - P_c) \tag{4}$$

Com o aumento na quantidade de dispositivos, tem-se mais colisões, devido ao número limitados de *slot* de acessos.

4.2. Proposta

Como apresentado nas Seções 1 e 3, os mecanismos (e.g., [Lee et al. 2011, Hasan et al. 2013, Lo et al. 2011]) para neutralizar a sobrecarga e o congestionamento apresentam-se como boas referências para o estudo do problema. Com exceção de [Jian et al. 2013], os mecanismos supracitados, quando analisados dentro de um contexto prático, apresentam alta complexidade de implantação. Em [Jian et al. 2013], as alterações são realizadas na camada MAC, especificamente no modo como o dispositivo calcula o *backoff* e no parâmetro que indica o nível de congestionamento na RAN. Contudo, a estratégia apresentada em [Jian et al. 2013] não considera na divisão das classes a quantidade de dispositivos em cada classe. Dessa forma, subutiliza recursos da rede alocando intervalos de *backoff* iguais para quantidades diferentes de dispositivos. Além disso, o algoritmo não estabelece priorização entre dispositivos de uma mesma classe.

4.2.1. Identificação do Nível de Congestionamento

Na proposta apresentada neste artigo, para auxiliar no controle da sobrecarga e congestionamento a estação base informa periodicamente para os dispositivos o nível de

congestionamento (P_{cong}) na rede. Classificados em baixo, médio e alto, o nível de congestionamento é baseado nos resultados apresentados em [Cheng et al. 2012]. Em [Cheng et al. 2012], os autores apresentam uma relação entre o número de solicitações recebidas, a probabilidade de colisão e a taxa de utilização dos *slots* para acesso aleatório por procedimento de RACH. Os resultados mostram que, para o nível de utilização máximo de aproximadamente 50% e a taxa de colisão de 20%, o número médio de solicitações é de 50%. Assim, para acima de 50 solicitações por *slot* o nível de congestionamento é considerado alto ($P_{cong} = 1$), e baixo (P_{cong} = 0) para valores abaixo de 25. Assim sendo, no presente trabalho o nível de congestionamento usado é classificado em baixo, médio e alto. A variação do nível de congestionamento (P_{cong}) e a relação com o número de solicitações de acesso recebidas estão representadas na Tabela 1.

Tabela 1. Níveis de congestionamento.

Nível	**Número de solicitações de acesso por *slot* de acesso**	P_{cong}
Baixo	< 25	P_{cong} = 0.0
Médio	> 25 e ≤ 50	P_{cong} = 0.5
Alto	> 50	P_{cong} = 1.0

4.2.2. Priorização entre Dispositivo

Na presente proposta os dispositivos são classificados em três tipos: H2H, M2M de alta prioridade e M2M de baixa prioridade [Lin et al. 2014]. Para garantir a priorização de acordo com o tipo de dispositivos, optou-se pela criação de classes de dispositivos. Os dispositivos da classe $i = 0$ apresentam P_s fixo e igual a p_{ac}. As demais classes são priorizadas segundo a probabilidade de acesso (P_{access}), de forma que, quanto menor a classe, maior a probabilidade de acesso. A probabilidade de transferência de solicitação de acesso (P_s) identifica quando o dispositivo pode transmitir uma solicitação de acesso para a estação base no (*RA-Slot*) durante o procedimento de RACH. Baseado na estratégia do ACB, antes de transmitir, o dispositivo gera um valor aleatório (X) uniformemente distribuído dentro do intervalo (0,P_s), sendo P_s dado pela Eq. 5, para todo $i \neq 0$.

$$P_s(i,t,L) = \begin{cases} p_{ac} & \text{para } P_{cong} = 0.0; \\ p_{ac} \times (((\frac{L}{t}) \times i) \times \alpha) & \text{para } P_{cong} = 0.5; \\ p_{ac} \times (((\frac{L}{t}) \times i) \times \alpha) & \text{para } P_{cong} = 1.0; \end{cases} . \quad (5)$$

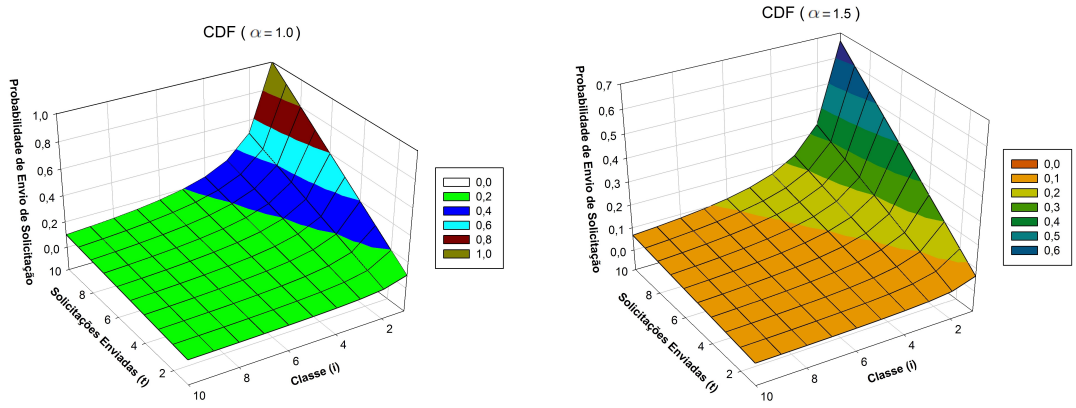
Sendo i utilizado para representar os tipos de dispositivos (H2H, M2M de baixa prioridade, M2M de alta prioridade), p_{ac} o parâmetro de bloqueio difundido pela estação base (eNB), L o número de retransmissões possíveis e t o número de tentativas de acesso realizadas pelo dispositivo. O parâmetro α define a dispersão entre as classes de acordo com o nível de congestionamento (P_{cong}). Pelos resultados obtidos através de simulações, os melhores valores obtidos para α, quando P_{cong} = 0.5 e P_{cong} = 1.0 são 1.0 e 1.5, respectivamente. O cálculo da probabilidade média de transferência de solicitação de acesso ($E[X]$) de um dado dispositivo k da classe i é dado por:

$$E[X] = \frac{1}{2} \times (0 + P_s(i_k, t_k, L)) \quad (6)$$

Baseado no funcionamento do ACB, o dispositivo pode transmitir uma solicitação de acesso durante o procedimento RACH quando $X \leq p_{ac}$, ou seja, com probabilidade de acesso $P_{access}(X \leq p_{ac})$. Para finalizar, a função (F_X) de densidade acumulada (*Cumulative Density Function* - CDF) do dispositivo k da classe i após t tentativas de acesso é dada por:

$$F_X(p_{ac}) = P(X \leq p_{ac}) = \frac{p_{ac}}{P_s} \tag{7}$$

Com base nas equações (6) e (7), é possível identificar que a probabilidade de envio do preâmbulo durante o acesso aleatório aumenta de acordo com a classe e o número de tentativas de acesso realizadas pelo dispositivo, como ilustrado na Figs. 2(a) e 2(b). As Figs. 2(a) e 2(b) também mostram que com o aumento do número de classes no sistema, tem-se a redução na probabilidade de envio de solicitação das classes menos prioritárias. Contudo, para a quantidade de classes consideradas neste trabalho, o algoritmo proposto garante probabilidade de envio do preâmbulo de aproximadamente 50% para a classe com baixa prioridade.



(a) Cálculo da CDF. (b) Cálculo da CDF.

Figura 2. Funções de Densidade Acumulada - CDFs (cf. Eq. 7)

Caso o valor gerado X seja maior do que p_{ac} $(X > p_{ac})$ o dispositivo fica em *backoff* por $T_{backoff}$, calculado individualmente por cada dispositivo através da equação:

$$T_{backoff}(i) = \begin{cases} 20ms & \text{para i} = 0; \\ 50ms \times i & \text{para i} \geq 1; \end{cases} \tag{8}$$

O uso do *backoff* faz-se necessário para evitar que os dispositivos realizem sucessivas tentativas de acesso, mesmo quando a rede está congestionada.

5. Ambiente de Simulação e Resultados Numéricos

Nesta seção, apresenta-se e analisa-se os resultados obtidos na simulação.

Tabela 2. Parâmetros utilizados nas simulações.

Parâmetros Gerais da Rede	
Bandwidth	5 MHz (25 RBs)
Amostras	10
Número de Estações Base (eNB)	1
Índice de Configuração do PRACH	6 (n = 2)
Número de Máximo de Retransmissões de Preâmbulo (L)	10
Códigos de Preâmbulo Disponíveis	52 $(64 - N)$, $N = 12$
Tempo de Janela de Resposta (RAR)	5 ms
Dispositivos H2H	{ 100,100,100,...,100 }
Dispositivos M2M de alta prioridade	{ 10,500,1000,...,4000}
Dispositivos M2M de baixa prioridade	{ 2,125,250,...,1000 }
Taxa de Chegada (H2H)	Poisson(λ), λ = 1/300
Taxa de Chegada (M2M)	Poisson(λ), λ = 1/900
Intervalo de Chegada	[0,...,1] s
Tempo de Simulação	5 s
Parâmetros do algoritmo PClasses	
i = 0 para dispositivos H2H, i = 1 para dispositivos M2M de alta prioridade, i = 2 para dispositivos M2M de baixa prioridade.	
$\alpha = 1$ para $P_{cong} = 0.5$, α = 1.5 para $P_{cong} = 1.0$ (cf. Table 1).	
Parâmetros do algoritmo Jain	
T = 67.0, C = 2, k = 1	

5.1. Indicadores Chave de Desempenho

Os dados obtidos através das simulações são comparados segundo os seguintes indicadores: tempo de acesso, probabilidade de bloqueio e número de acessos à rede realizados com sucesso. 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. A probabilidade de bloqueio é definida pela razão entre número de acessos realizados com sucesso e o número total de dispositivos requisitando acesso. Para finalizar, o número médio de acesso é dado pela média de acessos realizados com sucesso dentro do intervalo de simulação considerado.

5.2. Ambiente de Simulação

O simulador de rede baseado em eventos NS-3 foi utilizado para simulação da proposta apresentada neste trabalho [NS-3 2014]. Contudo, o acesso aleatório implementado no módulo LTE do NS-3 possui funcionalidades ainda não implementadas. Além disso, para o número de dispositivos M2M simulados, a performance do simulador é extremamente baixa. Assim, para o propósito deste trabalho, algumas funcionalidades do acesso aleatório foram implementadas no módulo LTE do NS-3.

Para a avaliação dos indicadores chave (PKIs) da proposta, três outras abordagens compatíveis com a arquitetura atual da rede LTE existentes na literatura foram implementadas. A primeira é o *Slotted Aloha*, que retrata o impacto nos dispositivos quando nenhum controle para o congestionamento é utilizado no procedimento de RACH. A segunda é o *backoff* específico, que atribui intervalos de *backoff* predeterminados para dispositivos H2H e M2M [ZTE 2011]. Devido a não priorização entre dispositivos M2M na proposta de *backoff* específico, a terceira abordagem implementada é a de [Jian et al. 2013]. Nela a priorização entre as classes de dispositivos ocorre através da atribuição de intervalos de *backoff* distintos entre as classes de dispositivos.

O cenário de simulação apresenta dispositivos H2H e M2M, sendo a quantidade de dispositivos H2H mantida constante e igual a 100. Os dispositivos M2M, como apresentado na Seção 4.2.2, são classificados em M2M de alta prioridade e M2M de baixa

prioridade. A quantidade de dispositivos M2M de baixa prioridade varia do seguinte modo: {10, 500, 1000, 1500, ... ,4000}, e a quantidade dos dispositivos M2M de alta prioridade: {2, 125, 250, ... , 1000}, ou seja, $\frac{1}{4}$ da quantidade dos dispositivos M2M de baixa prioridade. A taxa de chegada considerada para os dispositivos H2H e M2M segue a distribuição de Poisson e são respectivamente $\lambda_{H2H} = \frac{1}{300}$ e $\lambda_{M2M} = \frac{1}{900}$. A quantidade de códigos de preâmbulo disponíveis é (64-N), sendo N o número de preâmbulos alocados para o acesso livre de contenção e configurado para 12. Com a possibilidade de transmitir a cada 5ms, tem-se 2 *slots* de acesso disponíveis por *frame* LTE (10 ms), totalizando 200 RACH/s (1000/10 $\times$ 2). Para a largura de banda de 5 MHz, tem-se 25 blocos de recursos físicos (*Physical Resource Blocks* - PRBs) para cada 0,5 ms. Sabendo que um *slot* de acesso do RACH ocupa o equivalente a 6 PRBs no domínio da frequência e 1 ms no domínio do tempo, a estação base é capaz de atender até 4 (25/6) clientes por vez. Desta forma, em um cenário ideal, ou seja, sem colisões, 800 (200 RA-Slots/s $\times$ 4) dispositivos podem acessar a rede com sucesso no intervalo de 1 segundo. O intervalo de confiança utilizado foi de 95% e cada cenário (i.e, 112, 725, 1350, ... ,5100 dispositivos) foi simulado 10 vezes. As configurações acima estão sumarizadas na Tabela 2.

5.3. Mecanismo de Controle de Sobrecarga Simulados

As estratégias [Jian et al. 2013, 3GPP 2014b, 3GPP 2014a] apresentam em comum com o mecanismo proposto neste trabalho a viabilidade de implementação em um cenário prático de congestionamento nas redes LTE. Em [Jian et al. 2013], três técnicas são apresentas, porém, devido aos problemas já discutidos na Seção 3, a segunda técnica foi escolhida, uma vez que a terceira considera que a rede também está utilizando a técnica *Access Class Barring* (ACB) para controle do acesso dos dispositivos durante o procedimento de RACH. Como os autores em [Jian et al. 2013] não detalham ou fornecem informações sobre o nível de congestionamento utilizado, adotou-se a estratégia utilizada no presente trabalho que está descrita na Seção 4.2.1.

O *Slotted Aloha* não estabelece distinção entre os dispositivos, mas auxilia na análise e compreensão do problema do congestionamento. O intervalo de *backoff* considerado para o *Slotted Aloha* é de 20 ms [3GPP 2014a]. Para o Backoff Específico, nos cenários apresentados em [3GPP 2014a], o tempo máximo atribuído aos dispositivos M2M (960 ms) encontra-se dentro do intervalo de chegada dos dispositivos. Para manter esse comportamento, o valor máximo atribuído aos dispositivos M2M é 100 ms (Intervalo de chegada / Número máximo de retransmissões do preâmbulo).

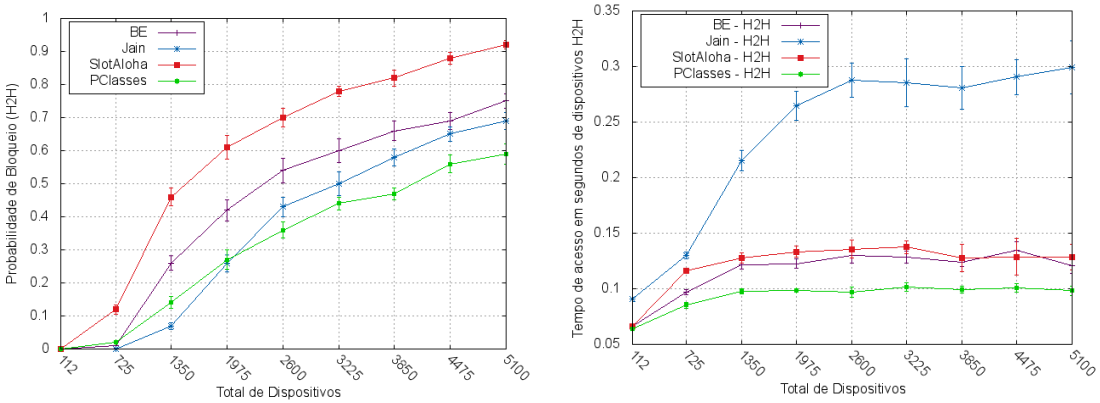
5.4. Resultados

Os algoritmos propostos em [Jian et al. 2013, 3GPP 2014b, 3GPP 2014a] serão referenciados respectivamente por “Jian”, “SlotAloha” e “BE”. O algoritmo proposto neste trabalho será referenciado por “PClasses”. Como apresentado na Seção 5.1, os indicadores (PKIs) são utilizados na análise dos seguintes aspectos das propostas: (i) controle do impacto sobre dispositivos H2H e (ii) priorização entre dispositivos M2M.

5.4.1. Controle do Impacto sobre Dispositivos H2H

A variação da probabilidade de bloqueio em relação a quantidade de dispositivos mostrada na Fig. 3(a) indica que para 5100 dispositivos o algoritmo PClasses apresenta melhor

performance em ralação a Jain e SloAloha de, respectivamente, 19% e 40% aproximadamente. Para 1350 dispositivos, percebe-se que o algoritmo de Jain apresenta performance máxima de aproximadamente 10% em relação ao PClasses. Contudo, com o aumento do número de dispositivos, tem-se uma redução na performance do algoritmo Jain. Entre o intervalo de 1800 a 2000 dispositivos percebe-se que há uma inversão da probabilidade de bloqueio, ou seja, PClasses apresenta probabilidade de acesso menor do que o algoritmo de Jain. Nos cenários com mais dispositivos a performance de PClasses é mantida e gradativamente aumentada.



(a) Probabilidade de Bloqueio. (b) Tempo Médio de Acesso.

Figura 3. Controle do Impacto Sobre Dispositivos H2H.

A inversão que ocorre próxima dos 1975 dispositivos está relacionada com o nível de congestionamento (P_{cong}), que foi detalhado na Seção 4.2.1. Para cenários com nível de congestionamento moderado o controle do acesso é menos rígido e consequentemente mais dispositivos M2M tentam acessar à rede. Consequentemente, tem-se o aumento do número de colisões e impacto no acesso do H2H.

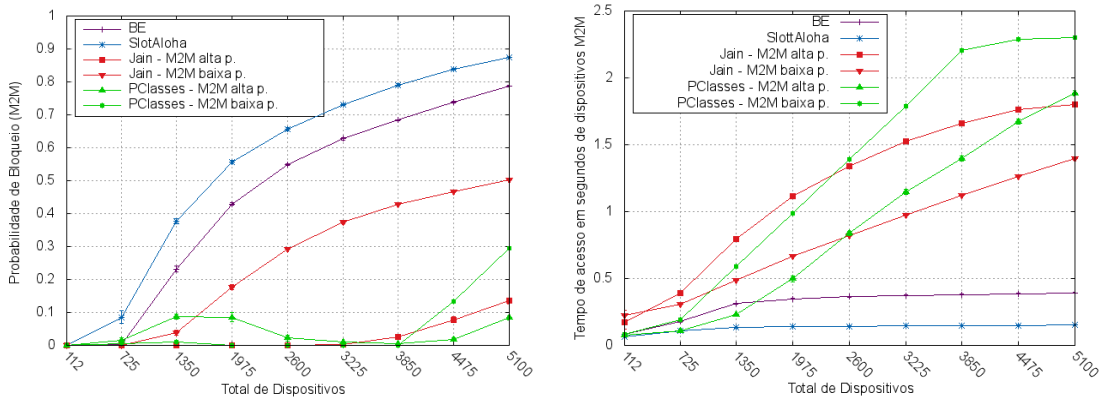
Como ilustrado na Fig. 3(b), o tempo médio de acesso permanece praticamente constante para todos os algoritmo simulados. Contudo, vale ressaltar que o cálculo do tempo médio de acesso considera somente os dispositivos que conseguiram acesso à rede com sucesso. Assim, o tempo médio de acesso dos dispositivos M2M permanece constante mas, em contrapartida, tem-se a diminuição do número de dispositivos que conseguem acessar a rede, como ilustrado na Fig. 3(a). Devido ao modo como o algoritmo Jain cria as classes de backoff, tem-se um tempo de acesso elevado para os dispositivos H2H.

5.4.2. Priorização entre Dispositivos M2M

Como apresentada na Fig. 4(a), a probabilidade de bloqueio aumenta com o número de dispositivos em todas as propostas analisadas. Os algoritmos SlotAloha e BE não estabelecem priorização entre os dispositivos M2M, penalizando igualmente todos os dispositivos M2M. Comparados entre si, a diferença existente entre o SlotAloha e o BE ocorre devido a diferença nos valores de *backoff*.

No SlotAloha os dispositivos M2M são dispersados em intervalos de tempo de 20 ms e no BE em intervalos de 100 ms. A variação no intervalo de *backoff* implica redução

na probabilidade de bloqueio e consequente variação na probabilidade de bloqueio. Os resultados apresentados por Jain são melhores do que o SlotAloha e BE, pois estes não apresentam priorização entre dispositivos M2M. Em relação ao PClasses, Jain apresenta uma melhor performance de aproximadamente (12%) na probabilidade de bloqueio entre dispositivos M2M de alta prioridade para cenários com aproximadamente 725 a 3530 dispositivos. Contudo, esse quadro se inverte quando o número de dispositivos M2M é superior a 3530. Em relação ao M2M de baixa prioridade a estratégia adotada em Jain comporta-se como o SlotAloha e o BE, ou seja, o tempo de acesso aumenta gradativamente com o aumento do número de dispositivos.



(a) Probabilidade de Bloqueio. (b) Tempo Médio de Acesso.

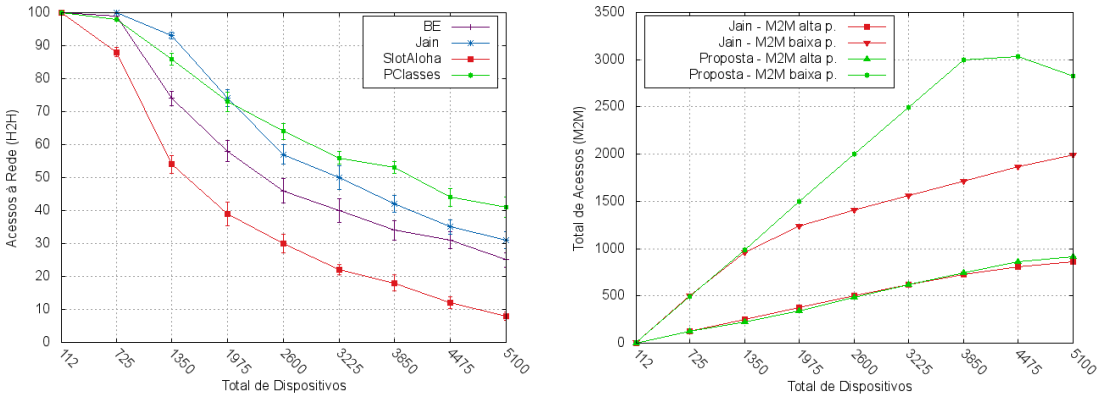
Figura 4. Priorização entre dispositivos M2M.

O algoritmo PClasses apresenta a mesma performance do algoritmo Jain quando comparado com o SlotAloha e BE em relação a priorização entre os dispositivos M2M. Percebe-se que o algoritmo PClasses excede o número de tentativas de acesso nos cenários menos congestionados, ou seja, aqueles em que há poucos dispositivos (e.g. Fig. 4(a) com 1350 dispositivos M2M). Porém, à medida que o nível de congestionamento aumenta o PClasses melhora. O comportamento apresentado pelo algoritmo PClasses para cenários com 1350 a 3850 dispositivos deve-se ao parâmetro (P_{cong}), pois este repercute na probabilidade de transmissão dos dispositivos, como apresentado nas equações (5) e (7).

5.4.3. Acessos Realizados com Sucesso

Através das Figs. 5(a) e 5(b) é possível verificar o impacto gerado nos dispositivos H2H com o aumento do número de dispositivos M2M. Os algoritmos Jain e PClasses apresentam comportamento parecido, contudo, percebe-se que PClasses converge para valores maiores do que Jain. Para os cenários em que o número de dispositivos é maior do que 1800, o algoritmo PClasses apresenta uma performance melhor do que o algoritmo Jain quanto ao tempo e número médio de acessos realizados com sucesso (cf. Fig. 3(b)).

Para os dispositivos M2M de alta prioridade o algoritmo PClasses apresenta vantagens quanto ao número e tempo médio de acessos em cenários com até aproximadamente 1350 dispositivos. Contudo, percebe-se que a partir de 1350 dispositivos o tempo médio de acesso do algoritmo PClasses é maior do que o algoritmo Jain, mas o algoritmo PClas-



(a) Dispositivos H2H. (b) Dispositivos M2M de alta e baixa prioridade.

Figura 5. Número Médio de Acessos Realizados com Sucesso.

ses mantém uma performance melhor do que o algoritmo de Jain quanto ao número médio de acessos.

6. Conclusões

A partir da distribuição dos dispositivos em classes, diversos cenários foram simulados. A análise dos resultados aponta que a variação na probabilidade de acesso e tempo de *backoff* como estratégia para distinção entre as classes podem mitigar o impacto do congestionamento ocasionado pelo excesso de dispositivos M2M na rede LTE. Também foi observado que o comportamento da estratégia empregada na identificação do nível de congestionamento pela estação base eleva o nível de restrição nos cenários mais congestionados (e.g. com mais de 4000 dispositivos), garantindo a priorização entre os dispositivos M2M e controlando o impacto sobre H2H. O uso de uma estratégia condizente com a realidade da rede LTE, mesmo com o impacto esperado, apresenta-se como uma boa opção para este estudo. A estratégia empregada para distinção das classes apresenta baixa complexidade de implementação, utilizando uma distribuição amplamente conhecida (Uniforme) e a técnica de *Backoff*.

Os resultados apresentados ajudam a entender a importância do emprego de mecanismo para mitigar o impacto da sobrecarga e do congestionamento na rede LTE ocasionado pelo excesso de dispositivos M2M. Neste sentido, o modo de priorização entre as classes apresentado neste trabalho apresenta resultados consistentes quando comparados com outros trabalhos que abordam a mesma problemática e que prezam por soluções próximas do contexto prático da rede LTE.

Referências

3GPP (2012). System improvements for machine-type communications. Technical report, 3GPP.

3GPP (2014a). Study on ran improvements for machine-type communication. Technical report, 3GPP.

3GPP (2014b). “evolved universal terrestrial radio access (e-utra); medium access control (mac) protocol specification,” 3rd generation partnership project (3gpp). Technical report, 3GPP.

Chang, Y., Zhou, C., and Bulakci, O. (2014). Coordinated random access management for network overload avoidance in cellular machine-to-machine communications. In *European Wireless 2014; 20th European Wireless Conference; Proceedings of*, pages 1–6. VDE.

Chen, J.-L., Hsieh, H.-C., and Larosa, Y. (2013). Congestion control optimization of m2m in lte networks. In *Advanced Communication Technology (ICACT), 2013 15th International Conference on*, pages 823–827.

Cheng, M.-Y., Lin, G.-Y., Wei, H.-Y., and Hsu, C.-C. (2012). Performance evaluation of radio access network overloading from machine type communications in lte-a networks. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 248–252. IEEE.

Hasan, M., Hossain, E., and Niyato, D. (2013). Random access for machine-to-machine communication in lte-advanced networks: issues and approaches. *Communications Magazine, IEEE*, 51(6).

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.

Laya, A., Alonso, L., and Alonso-Zarate, J. (2014). Is the random access channel of lte and lte-a suitable for m2m communications? a survey of alternatives. *Communications Surveys Tutorials, IEEE*, 16(1):4–16.

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.

Maia, A. M., de Castro, M. F., Vieira, D., and Ghamri-Doudane, Y. (2014). Um mecanismo para escalonamento de pacotes no uplink da rede lte no contexto da comunicaç ao máquina-a-máquina. In *Proceedings of the 32th Brazilian Symposium on Computer Networks (SBRC 2014)*.

NS-3 (2014). *The network simulator ns-3*.

Phuyal, U., Koc, A. T., Fong, M.-H., and Vannithamby, R. (2012). Controlling access overload and signaling congestion in m2m networks. In *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on*, pages 591–595. IEEE.

Zheng, K., Ou, S., Alonso-Zarate, J., Dohler, M., Liu, F., and Zhu, H. (2014). Challenges of massive access in highly dense lte-advanced networks with machine-to-machine communications. *Wireless Communications, IEEE*, 21(3):12–18.

ZTE (2011). Backoff enhancements for ran overload control. Technical report, 3GPP.

MINEIRO: Um Mecanismo de Incentivo para Aplicações em Redes Oportunísticas*

Vinícius F. S. Mota[1,2], Daniel F. Macedo[1], Yacine Ghamri-Doudane[3], José Marcos Silva Nogueira[1]

[1] Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Belo Horizonte – Brazil

[2]LIGM – Université Paris Est
Paris – France

[3]L3i Lab – Université de La Rochelle
La Rochelle – France

{vfsmota, damacedo, jmarcos}@dcc.ufmg.br,yacine.ghamri@univ-lr.fr

Abstract. *In the recent years, the research community has proposed several protocols and applications for opportunistic networking. These protocols require that all users are willing to cooperate with the network. However, users could have selfish behavior, which can lead to degradations in the network performance. In this work, we propose an incentive mechanism to improve routing, called MINEIRO, which aims to detect and avoid selfish nodes based on the source of the messages. We demonstrate that our algorithm leads to bayeasian equilibrium. Moreover, we analyzed the impact of selfishness in opportunistic networks and showed that without an incentive mechanism, the delivery ratio is constant for up to 60% of nodes with selfish behavior in a network with random node mobility. Meanwhile, our proposal incentivizes rational users to forward messages regardless of the amount of selfish nodes.*

Resumo. *Nos últimos anos, a comunidade científica propôs diversos protocolos e aplicativos para redes oportunísticas. Estes protocolos requerem que todos os usuários estejam dispostos a cooperar com a rede. No entanto, os usuários podem ter comportamento egoísta, o que pode afetar o desempenho da rede. Neste trabalho, propomos um mecanismo de incentivo para melhoria do roteamento em redes oportunísticas, chamado MINEIRO, que tem como objetivo detectar e evitar os nós egoístas com base na origem das mensagens recebidas. O MINEIRO utiliza apenas informações locais. Demonstramos sob quais condições nosso algoritmo atinge o equilíbrio bayesiano em um jogo dinâmico. Além disso, analisamos o impacto do egoísmo em redes oportunísticas e mostramos que, sem nenhum mecanismo, a taxa de entrega é constante para até 60% de nós com comportamento egoísta em redes com mobilidade aleatória. Nossa proposta, por sua vez, incentiva os usuários racionais a encaminharem mensagens em qualquer quantidade de nós egoístas.*

*Os autores gostariam de agradecer à CAPES, COFECUB, CNPq e FAPEMIG pelo apoio financeiro parcial a este trabalho.

1. Introdução

Smartphones, *tablets* e outros dispositivos com capacidade de comunicação tornaram-se muito populares nos dias de hoje. A capacidade de armazenamento e processamento desses dispositivos têm aumentado significativamente, bem como suas tecnologias de comunicação têm ficado cada vez melhores, as quais incluem Bluetooth, Wi-Fi e as atuais redes LTE, popularmente chamadas de redes 4G.

Devido à explosão de tráfego em redes estruturadas e à dependência da existência de uma rede comum entre fonte e destino das mensagens, tem-se observado uma tendência de uso de comunicação *ad hoc* entre os dispositivos. Na comunicação *ad hoc*, os dispositivos encaminham mensagens entre si, sem necessidade de qualquer infra-estrutura. Como exemplo, o consórcio 3GPP está considerando o uso de tecnologia *ad hoc* em versões futuras do LTE [3GPPP 2013], e a WiFi Alliance padronizou o WiFi Direct [2014]. Tais tecnologias podem ser empregadas para a transmissão de dados com múltiplos saltos, utilizando um sistema de repasse oportunístico, no qual os dados são repassados sempre que ocorre um "contato" entre um par de nós [Fall 2003].

O repasse oportunístico depende de algoritmos de roteamento adequados. São descritos diversos algoritmos de roteamento na literatura, que se diferem pela decisão dos nós de repassarem ou não as mensagens quando entrarem em contato com outros nós [Mota et al. 2014]. Uma suposição comum é que todos os nós são confiáveis e irão participar da rede, isto é, irão encaminhar corretamente mensagens vindas de outros nós. Esta suposição é válida em redes nas quais todos os nós cooperam entre si para alcançar uma determinada tarefa, por exemplo, em uma rede de sensores sem fio para monitoramento ambiental ou em uma rede criada para uma missão de resgate após um desastre. No entanto, em aplicações nas quais os indivíduos executam tarefas não relacionadas, como por exemplo em uma rede LTE, o comportamento humano tende ao egoísmo, ou seja, o indivíduo deseja usufruir dos benefícios da rede, porém não deseja cooperar com a mesma [Miao et al. 2013]. Por essa razão, o comportamento humano deve ser considerado pelos algoritmos de roteamento.

Existem diversos motivos para o comportamento egoísta em redes oportunísticas, entre os quais podemos citar: a necessidade de economia de energia do dispositivo; economizar gastos de transmissão de dados; questões de privacidade [Lee and Hoh 2010]. A Figura 1 representa os custos envolvidos ($C(x)$) na tomada de decisão do nó B quando uma mensagem originária de A deve ser repassada a C. Porém, não há garantias que B irá encaminhar a mensagem para C, pois o nó B pode considerar o custo (por exemplo o consumo de energia) para receber e retransmitir a mensagem alto. Por isto, mecanismos de incentivo visam aumentar a probabilidade dos nós cooperarem efetivamente com a rede.

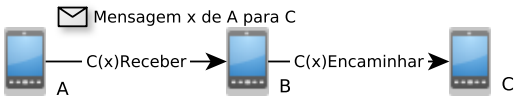


Figura 1. Um exemplo de transmissão em uma rede ad hoc. O dispositivo *B* deve considerar o custo para receber e retransmitir a mensagem *x*.

Neste trabalho propomos um mecanismo de incentivo ao repasse de dados batizado de MINEIRO - *Message-based INcentive mechanism for End-user Improvement of Routing Opportunities*. O MINEIRO constrói uma tabela de reputação dos nós baseada na origem da mensagem recebida e qual nó está encaminhando a mesma. Esta tabela é construída com base apenas em informações locais, evitando assim entidades certificadoras centrais. Utilizando a Fig. 1 como exemplo, se o dispositivo *B* retransmite a mensagem *x* de *A* para *C*, então o MINEIRO em *C* aumentará a reputação do dispositivo *B*.

Apresentamos uma modelagem de nossa proposta baseada em teoria dos jogos. Por meio desta modelagem, mostramos para quais parâmetros o MINEIRO atinge o equilíbrio Bayesiano, o qual especifica uma distribuição de probabilidades sobre as ações de cada tipo de jogador de forma que cada um desses tipos maximize seus resultados [Gaoning et al. 2010].

Além disto, analisamos o impacto do comportamento egoísta no algoritmo de encaminhamento Epidêmico [Vahdat and Becker 2000] em dois cenários de mobilidade: com mobilidade aleatória e em um modelo de mobilidade baseado em registros de movimentação humana. A avaliação mostra que o MINEIRO apresenta melhores resultados em relação às métricas de taxa de entrega e atraso quando considerado que os nós podem ser egoístas, mas são racionais. Isto é, cooperam com a rede quando necessário para que possam aumentar o seu desempenho. As contribuições deste trabalho são sumarizadas a seguir:

- Proposta de um mecanismo de incentivo baseado em informações locais para a construção de uma tabela de reputação.
- Modelagem do mecanismo de incentivo como um jogo *Bayesiano*.
- Análise do impacto do comportamento egoísta em redes oportunísticas considerando dois tipos de egoísmo: *i*) Nós não reencaminham as mensagens, porém as mantêm em seu *buffer*; *ii*) Nós descartam todas as mensagens.

O restante deste artigo está organizado da seguinte forma: Na Seção 2 contextualizamos o trabalho e apresentamos a revisão da literatura relacionada aos mecanismos de incentivo. Apresentamos o nosso mecanismo de incentivo e de detecção de egoísmo na Seção 3. Uma abordagem de teoria dos jogos é utilizada para mostrar sob quais condições nosso algoritmo atinge o equilíbrio Bayesiano na Seção 4. A avaliação de desempenho e os resultados obtidos são apresentados na Seção 5. Por fim, as considerações finais são apresentadas na Seção 6.

2. Comportamentos Humanos e Mecanismos de Incentivo

O comportamento social do ser humano, como egoísmo e altruísmo, tem sido extensivamente estudado na filosofia, psicologia, economia e no contexto da ciência da computação. Neste último, ele tem sido estudado em sistemas par-a-par, redes *ad hoc* e redes oportunísticas [Miao et al. 2013]. O egoísmo pode ser definido como o ato de beneficiar somente a si próprio ao invés do outro. Por outro lado, o comportamento altruísta ou generoso leva a pessoa a favorecer outros ao invés de si próprio. Levine demonstrou experimentalmente com um jogo simples que existem pessoas egoístas e altruístas em qualquer população [Levine 1998]. A cooperação ocorre quando um indivíduo dedica um esforço, que implica em um custo, em alguma atividade coletiva de forma que outros participantes da atividade recebam um benefício maior do que o custo [Bowles and Gintis 2003].

A diferença entre cooperação e altruísmo é que na cooperação o indivíduo espera algum benefício por ter cooperado.

O comportamento egoísta tem sido um gargalo para aplicações na Internet que dependem de colaboração do usuário. Por exemplo, no Gnutella, um popular sistema de compartilhamento de arquivos par-a-par de alguns anos atrás, 70% dos usuários não compartilhavam nenhum arquivo e 50% dos pedaços de arquivos transmitidos na rede eram enviados pelos 1% dos usuários mais ativos [Adar and Huberman 2000].

Para redes móveis ad hoc, Crowcroft *et al.* modelaram um mecanismo de incentivo com base no uso da banda e energia de cada nó [Crowcroft et al. 2004]. Neste modelo, cada nó recebe um crédito inicial. Um usuário aumenta o seu crédito ao encaminhar o tráfego de outros nós e diminui o crédito com base no custo da energia e consumo da largura de banda do tráfego criado.

No primeiro estudo sobre o impacto do egoísmo em redes oportunísticas, Panagakis *et al.* mostram que a taxa de entrega diminui linearmente com a quantidade de nós egoístas [Panagakis et al. 2007]. Hui *et al.* analisaram o impacto do egoísmo e do altruísmo usando vários *traces* reais de contato e de trânsito [Hui et al. 2009]. Em todos os cenários analisados, a rede formada somente por nós altruístas obteve um atraso 20% inferior ao atraso obtido quando houveram nós egoístas. Portanto, mecanismos de incentivo visam engajar os nós a cooperarem entre si, atenuando o impacto do egoísmo no desempenho da rede.

Os mecanismos de incentivo em redes oportunísticas podem ser classificados como: baseados em permutação - *barter-based* (também conhecido como *tit-for-tat*); baseados em créditos; e baseados em reputação [Miao et al. 2013].

Nas estratégias *baseadas em permutação*, os nós trocam a mesma quantidade de mensagens entre si, proporcionando assim um equilíbrio na utilização dos recursos entre os nós [Buttyán et al. 2010]. No entanto, esta estratégia impõe um limite forte sobre o número de mensagens trocadas entre os nós.

Nas estratégias *baseadas em créditos*, os usuários recebem um pagamento para incentivar a colaboração. O pagamento pode ser feito por cada mensagem encaminhada ou somente quando a mensagem alcança seu destino. Em [Yang et al. 2012], os autores propuseram dois mecanismos de incentivo à base de crédito para redes de sensoriamento participativas. O objetivo é incentivar os usuários a utilizarem seus *smartphones* para coletar e analisar dados de forma descentralizada. Como desvantagem, os mecanismos baseados em crédito exigem um coordenador central para validar o pagamento.

Nas estratégias *baseadas em reputação*, os nós visam detectar e evitar outros nós egoístas, classificando-os de acordo com algumas métricas. Por exemplo, no mecanismo *Ironman*, são atribuídas reputações positivas a todos os nós que participaram do encaminhamento de uma mensagem da origem ao destino [Bigwood and Henderson 2011]. Um mecanismo semelhante é o *MobiGame* [Wei et al. 2011], o qual funciona como se segue: Um nó A cria uma mensagem x para um nó C, e em seguida A encaminha x para o nó B. B, por sua vez, a reencaminha para C. Após recebê-la, o nó C envia para B um recibo autenticado de retransmissão da mensagem x. Finalmente, quando o nó A reencontrar o nó B, que estará com o recibo de retransmissão de x, o nó A aumentará a reputação de B.

Todos os mecanismos citados anteriormente requerem uma entidade central para garantir que um nó esteja dizendo a verdade sobre outro nó ou para certificar a autenticidade dos recibos de transmissão das mensagens entregues. Quando possível, o uso de entidade central facilita tarefas como autenticação e gerenciamento da rede. No entanto, a utilização de uma entidade central pode não ser viável em redes oportunísticas, uma vez que nós não certificados também deveriam participar da rede para aumentar as chances de repasse da mensagem. Assim, mecanismos de incentivo para as redes oportunísticas devem empregar apenas métodos que envolvam somente interações entre os participantes da rede.

Com base nesta observação, apresentamos um novo mecanismo de incentivo, batizado de MINEIRO, no qual um nó atribui reputação a outros nós conforme eles retransmitam mensagens de outros nós ou apenas as mensagens de si mesmos. Quanto mais mensagens de terceiros, maior a reputação atribuída. A principal diferença da nossa proposta em relação ao estado da arte é que nosso mecanismo classifica como egoístas todos os usuários que enviam mais mensagens próprias do que mensagens de terceiros. Deste modo, evitamos depender de informação sobre a reputação de um nó vinda de outros nós, o que requer mecanismos de autenticação, algo complexo em redes abertas.

3. Um Mecanismo de Incentivo Baseado na Origem da Mensagem

Nesta seção apresentamos o MINEIRO[1], ***M**essage-based **IN**centive mechanism for **E**nd-user **I**mprovement of **R**outing **O**pportunities* em redes oportunísticas. Primeiramente, apresentaremos o modelo de rede e os pressupostos exigidos pela nossa proposta. Em seguida, apresentaremos o nosso algoritmo.

3.1. Modelo de Rede

Uma rede oportunística pode ser modelada como um grafo de alcance temporal, ou grafo temporal [Whitbeck et al. 2012]. Neste tipo de rede não há garantias que exista um caminho fim-a-fim entre dois nós (u, v) em um determinado momento t. Porém, este caminho pode ser obtido quando considerado um intervalo entre t e $t + \delta$, no qual δ representa o limite superior do tempo para construção de um caminho entre dois nós. Whitbeck *et al.* definem como *jornada* o caminho construído durante o intervalo $[t, t + \delta]$.

Consideramos uma rede oportunística modelada como um grafo temporal $\mathcal{G}_\delta(V, A)$, sendo V o conjunto de vértices representando os nós da rede e A, o conjunto de arestas representando os enlaces entre os nós da rede. Devido à mobilidade, as arestas (enlaces) são adicionadas ou removidas ao grafo $\mathcal{G}_\delta$ do instante t até o tempo $t+\delta$. Existe uma jornada $\mathcal{J}$ de *u* para *v* se forem adicionadas arestas ao grafo $\mathcal{G}_\delta$, de tal forma que, o tempo total t' para *u* alcançar *v* satisfaça à equação $t' \leq t + \delta$.

Neste trabalho estendemos o modelo de Whitbeck para considerar o comportamento dos usuários. Consideramos que cada nó $v \in V$ possui uma variável contínua $\mathcal{R} = [0, 1]$ representando o comportamento de v ao transmitir as mensagens armazenadas em seu *buffer*. Um nó altruísta, $\mathcal{R} = 1$, retransmite todas as mensagens. Enquanto um nó egoísta, $\mathcal{R} = 0$, encaminha somente as mensagens próprias. Portanto, a proba-

[1]O acrônimo MINEIRO também é uma homenagem ao povo de Minas Gerais, conhecido por sua hospitalidade e por sempre darem um voto de confiança a estranhos.

bilidade de entrega de mensagens de u para v empregando uma jornada $\mathcal{J}_p$ é dada por $P(\mathcal{J}_p) = \prod_{f \in \mathcal{J}_p} \mathcal{R}_f$, sendo f os nós intermediários.

3.2. O Mecanismo de Incentivo Proposto

O objetivo do MINEIRO é classificar os nós de acordo com a origem das mensagens que os mesmos transmitem, para, desta forma, reconhecer e evitar os nós egoístas. Como consequência, para aumentar a probabilidade de terem suas mensagens entregues ao destino, os nós deverão encaminhar mensagens de outros nós.

O Algoritmo 1 descreve o nosso mecanismo de incentivo. O MINEIRO pode ser utilizado em conjunto com qualquer algoritmo de roteamento oportunístico, sendo necessário apenas que o roteamento envie para o MINEIRO uma lista contendo as informações sobre os remetentes das mensagens recebidas pelo nó u que foram encaminhadas por v (linha 1). O MINEIRO segue uma regra de prova de confiança: quando um par de nós, u e v, se encontram pela primeira vez, ambos os nós assumem uma reputação inicial maior que zero em relação ao outro e marcam a data e hora em que o evento ocorreu (linhas 2-4). Em seguida, e nos próximos encontros entre u e v, o nó u analisará as mensagens que o nó v deseja encaminhar. O nó u aumenta a reputação do nó v se a origem da mensagem for diferente de v (linhas 7-8). Caso contrário, o nó u diminui a reputação de v, isto é, quando v encaminha suas próprias mensagens (linhas 10-11). A tabela de reputação não sofre alterações se a mensagem é de v para u, visto que neste caso v deseja entregar uma mensagem própria, porém u tem interesse em recebê-la.

Após a atualização da tabela de reputação ($\mathcal{R}$), o MINEIRO classificará como egoísta todos os nós com reputação igual a zero, e mensagens provenientes destes nós serão rejeitadas, com exceção de quando a mensagem transmitida tem o nó que a recebeu como destino. Um nó com reputação positiva maior que zero terá sempre suas mensagens encaminhadas.

Algorithm 1 MINEIRO

1: **Input:** Nó u recebe lista das mensagens M encaminhadas por v
2: $T_v = now()$; {Atualizando data e hora}
3: **if** $\mathcal{R}_v \notin R$ **then**
4: $\quad \mathcal{R}_v = r_{init}$;
5: **end if**
6: **for all** $M_j \in M$ **do**
7: $\quad$ **if** $origem(M_j) \neq v$ **then**
8: $\quad\quad \mathcal{R}_v = min(\mathcal{R}_v + \Delta_{incremento}, 1)$;
9: $\quad$ **else**
10: $\quad\quad$ **if** $destino(M_j) \neq u \wedge origem(M_j) = v$ **then**
11: $\quad\quad\quad \mathcal{R}_v = max(\mathcal{R}_v - \Delta_{decremento}, 0)$;
12: $\quad\quad$ **end if**
13: $\quad$ **end if**
14: **end for**

Portanto, após a execução do algoritmo, o nó u tenta aproximar o valor da reputação de um nó v em relação ao comportamento do mesmo:

- **Altruísta -** $\mathcal{R}_v = 1$: O nó v encaminha todas as mensagens de todos os nós.

- **Racional -** $\mathcal{R}_v =]0, 1[$: O nó v rejeita as mensagens de nós egoístas.
- **Egoísta -** $\mathcal{R}_v = 0$: O nó rejeita todas as mensagens de todos os nós. Neste caso, o nó u rejeitará todas as mensagens encaminhadas por v.

Como um nó sempre recebe uma reputação inicial, o algoritmo permite que qualquer nó tenha oportunidade de transmitir suas próprias mensagens em um primeiro encontro. Um valor inicial alto da reputação faz com que o algoritmo demore mais a reconhecer um nó como egoísta, enquanto um valor muito baixo pode levar um nó a ser considerado erroneamente como um nó egoísta. O nó não encaminha mensagens enviadas por um nó v que seja considerado egoísta em suas tabelas de reputação, isto é, $\mathcal{R}_v = 0$. Consequentemente, isto diminui a probabilidade de entrega de mensagens de v.

O MINEIRO difere de outras propostas devido ao fato de não ser necessário que um nó tenha conhecimento prévio de outros nós ou que seja feita a troca de chaves públicas. O algoritmo evita entidades centrais para garantir a reputação de outros nós, uma vez que o tempo para atingir um terceiro nó em uma rede oportunística pode ser muito alto. Isto faz com que o cálculo de reputação seja mais rápido, uma vez que ele se baseia apenas em informações locais. No entanto, ressaltamos o compromisso que existe entre calcular a reputação rapidamente e calculá-la de maneira precisa, visto que um par de nós pode se encontrar somente uma vez durante o tempo de vida da rede.

O MINEIRO visa incentivar um comportamento racional nos nós. Enquanto o comportamento egoísta diminui a probabilidade de entrega de mensagens para o próprio nó, o comportamento puramente altruísta pode consumir os recursos de um nó rapidamente. Portanto, o comportamento racional faz com que um nó coopere com a rede a fim de melhorar sua própria qualidade de serviço e ao mesmo tempo economize recursos evitando os nós egoístas. Na próxima seção modelamos o MINEIRO como um jogo bayesiano e mostramos que encaminhar mensagens de terceiros é a melhor estratégia que um nó pode tomar para que ele aumente a própria reputação na rede e, por consequência, a probabilidade de ter suas mensagens entregues.

4. Modelagem Utilizando Teoria dos Jogos

A teoria dos jogos modela as situações em que o resultado de um participante de um sistema é afetado por sua decisão e pelas decisões tomadas por todos os outros participantes com quem esteja interagindo [Gaoning et al. 2010]. Nesta seção, daremos uma breve introdução sobre conceitos básicos de teoria dos jogos e modelaremos o nosso algoritmo como um jogo Bayesiano.

4.1. Fundamentos Básicos

Os participantes de um jogo são chamados de *jogadores*, enquanto que suas decisões são conhecidas como *movimentos* ou *estratégias* (usaremos o termo *estratégia* no restante do texto). Um jogador recebe um *pagamento* em função da sua estratégia e daquela escolhida pelos outros jogadores. Formalmente, um jogo consiste em um conjunto de jogadores, no qual cada jogador i seleciona uma estratégia $s_i \in S$. O conjunto das estratégias selecionadas por todos os jogadores é denotado por um tupla $P_i = (s_{i=1}, ..., s_n)$, e p_i é o pagamento a cada jogador, sendo p_i uma função de P_i.

As situações nas quais os jogadores possuem informações privadas e desconhecidas por outros jogadores são modeladas por meio de jogos de *informações incompletas*,

também chamados *jogos Bayesianos*. O envio de mensagens em redes oportunísticas é um exemplo de jogo Bayesiano, uma vez que um nó (um jogador) sabe seu comportamento, se será egoísta ou não, mas não sabe como o outro nó irá agir. Neste caso, um nó somente associa uma probabilidade em relação ao comportamento do outro nó. Três conceitos fundamentais na teoria dos jogos são necessários antes de apresentarmos nossa modelagem: *Melhor Resposta*, *Equilíbrio de Nash* e o *Equilíbrio Bayesiano*.

A *melhor resposta* é a estratégia escolhida por um jogador, que maximize seu pagamento independentemente da estratégia escolhida por outros jogadores. Formalmente, considerando dois jogadores (i, j):

$$P_i(s_i, s_j) \geq P_i(s'_i, s_j) \forall s' \in S$$

O *equilíbrio de Nash* é alcançado quando todos os jogadores escolhem estratégias que representam suas melhores respostas. A ideia por trás desse equilíbrio está no fato de que se um jogador escolher a estratégia mais segura, em que irá ganhar mais pontos independentemente das escolhas dos outros jogadores, então os outros jogadores irão fazer o mesmo.

Finalmente, O *equilíbrio Bayesiano* é a distribuição das probabilidades para cada tipo de estratégia para que cada jogador maximize seus pagamentos. O equilíbrio baseia-se na probabilidade de um jogador de um dado tipo escolher determinada estratégia.

4.2. MINEIRO como um Jogo Bayesiano

Consideramos como jogadores os participantes de uma rede oportunística. Estes jogadores podem apresentar dois comportamentos distintos: Serem egoístas ou serem cooperativos. O comportamento de um jogador não é conhecido *a priori* pelos outros jogadores. O MINEIRO permite a um jogador assumir que um outro jogador é egoísta ou não após uma quantidade de iterações entre os mesmos. Contudo, um jogador não sabe como o outro o enxerga, isto é, se o vê como egoísta ou não. Portanto, a racionalidade deverá fazer com que um jogador contribua com a rede visando que os outros jogadores o enxerguem com uma boa reputação para que suas mensagens sejam enviadas.

Modelamos o MINEIRO como um jogo bayesiano para analisar os valores de racionalidade $\mathcal{R}$ e quais parâmetros do algoritmo induzem ao equilíbrio bayesiano. O modelo considera o que ocorre quando um nó da rede encontra com outro. Assim, o jogo é modelado com somente dois jogadores. Formalmente:

Considere o jogo bayesiano $\mathcal{G}_{MINEIRO} = \langle \mathcal{N}, \mathcal{T}, \mathcal{A}, \mathcal{Q}, \mathcal{U} \rangle$:

- $\mathcal{N} = \{n_1, n_2\}$, o conjunto de jogadores, representando os participantes da rede.
- $\mathcal{T} = \{Eg : \text{Egoísta}, Co : Colaborador\}$, o conjunto de tipos de jogadores.
- $\mathcal{A} = \{Pr : \text{Enviar mensagem própria}, Te : \text{Enviar mensagem de terceiros}\}$, o conjunto de estratégias que cada jogador pode tomar.
- $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$, onde $\mathcal{Q}_k = [0, 1]$, representa a distribuição da probabilidade de acordo com o tipo do jogador. Assumimos $p(\mathcal{T}_1 = Eg) = \mathcal{R}$, logo $p(\mathcal{T}_1 = Co) = 1 - \mathcal{R}$
- $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2$, o conjunto de pagamento recebido pelos nós *k* baseado no tipo que pertencem e nas estratégias escolhidas.

	$\mathcal{R}_1$	
Eg	Pr	Te
Pr	$I-\Delta_{dec}, I-\Delta_{dec}$	$I-\Delta_{dec}, I+\Delta_{inc}$
Te	$0, I-\Delta_{dec}$	$0, I+\Delta_{inc}$

	$1-\mathcal{R}_1$	
Co	Pr	Te
Pr	$I-\Delta_{dec}, I-\Delta_{dec}$	$I-\Delta_{dec}, I+\Delta_{inc}$
Te	$I+\Delta_{inc}, I-\Delta_{dec}$	$I+\Delta_{inc}, I+\Delta_{inc}$

Tabela 1. Matriz de utilidades do jogo. Cada entrada (u_1, u_2) representa o pagamento (reputação) que cada jogador n_k recebe em relação à sua estratégia e à estratégia adotada pelo outro após o encontro do jogador n_1 com n_2.

A Tabela 1 representa a matriz de utilidades do jogo. As linhas da matriz representam a estratégia do jogador 1, enquanto as colunas representam a estratégia do jogador 2. Cada tupla (u_1, u_2) define os pagamentos $\mathcal{U}_k$ para cada conjunto de tipo e estratégia do nó n_k, sendo $\mathcal{R}_1$ a probabilidade do nó n_1 ser do tipo *Egoísta* e $1-\mathcal{R}_1$ a probabilidade do mesmo nó ser *Colaborador*. A partir desta tabela, derivamos o Teorema 1.

Teorema 1 *Para o comportamento de cooperação ser motivado é necessário que* $\Delta_{inc} \geq \Delta_{dec}$ *e que* $\mathcal{R}_k < \frac{\Delta_{inc}+\Delta_{dec}}{I+\Delta_{inc}}$*, sendo (Colaborador:enviar mensagem de terceiros, enviar mensagem terceiro) um equilíbrio bayesiano para o MINEIRO.*

Para demonstrarmos o Teorema 1, considere o encontro dos jogadores n_1 e n_2, onde cada jogador possui em seu *buffer* um conjunto de mensagens próprias e mensagens de terceiros, das quais nenhum dos jogadores são os destinos. Seja I o pagamento inicial que cada nó recebe quando não se conhecem, igual a r_{init}, Δ_{inc} e Δ_{dec} os parâmetros do MINEIRO referentes ao incremento e decremento de reputação, respectivamente.

A estratégia *enviar mensagem de terceiros* (Te) é estritamente dominante para o jogador n_2, se satisfeita a equação $I+\Delta_{inc} \geq I-\Delta_{dec}, \forall(\Delta_{inc} \geq \Delta_{dec})$.

Sejam $\sigma_1(Pr)$ e $\sigma_1(Te)$ os perfis de estratégias mistas que representam as probabilidades do jogador n_1 escolher a estratégia *enviar mensagem própria* ou a estratégia *enviar mensagem de terceiros*, respectivamente.

$$\sigma_1(Pr) = \sigma_1(Pr|Eg)+\sigma_1(Pr|Co) = \mathcal{R}_1(I-\Delta_{dec})+(1-\mathcal{R}_1)(I-\Delta_{dec}) = (I-\Delta_{dec})$$

$$\sigma_1(Te) = \sigma_1(Te|Eg) + \sigma_1(Te|Co) = (1-\mathcal{R}_1)(I+\Delta_{inc})$$

Como desejamos saber em quais condições a estratégia *enviar mensagens de terceiros* é melhor para o jogador 1, então devemos encontrar em quais situações $\sigma_1(Te) > \sigma_1(Pr)$. Para que $\mathcal{R}_1$ satisfaça a equação $\sigma_1(Te) > \sigma_1(Pr)$, temos que:

$$0 < \mathcal{R}_1 < \frac{\Delta_{inc}+\Delta_{dec}}{I+\Delta_{inc}}$$

Nestas condições, n_1 ser colaborador e enviar mensagem de terceiros e n_2 enviar mensagem de terceiros define um perfil de estratégia que leva ao equilíbrio bayesiano.

5. Avaliação de Desempenho

Avaliamos o desempenho de nossa proposta utilizando o *Opportunistic Networking Evaluator* (ONE) [Keränen et al. 2009]. ONE é um simulador de eventos discretos para redes oportunísticas. No ONE uma mensagem é totalmente entregue ou descartada, sem fragmentação.

5.1. Configurações da Simulação

A rede simulada possui 50 nós em uma área de 1000m x 1000m. O tempo simulado foi de 28800s. Os nós são equipados com interfaces 802.11 com um alcance de rádio de

50 metros e largura de banda 10Mbps. Estas configurações representam usuários em um shopping, por exemplo. A cada intervalo de [10-50]s é gerada uma mensagem para um par origem-destino escolhido aleatoriamente até o tempo de 20000s.

Os nós utilizam o protocolo de encaminhamento Epidêmico [Vahdat and Becker 2000]. Ressaltamos que o MINEIRO pode ser aplicado sobre qualquer protocolo de roteamento. No entanto, o Epidêmico pode replicar uma mensagem entre todos os nós da rede e, com isto, apresenta melhores resultados em relação à taxa de entrega e atraso quando comparado a outros protocolos. Comparamos o MINEIRO em relação a duas outras políticas:

- **Sem Incentivo**: Algoritmo Epidêmico sem qualquer política de incentivo.
- **Barter [Buttyán et al. 2010]**: Os nós trocam somente a mesma quantidade de mensagens.

Quando não há incentivo, o comportamento dos nós é *egoísta*. Quando há uma política de incentivo, o comportamento dos nós é *racional* de acordo com o mecanismo de incentivo utilizado.

Consideramos dois modelos de mobilidade:

- **Random Waypoint model** (RWP), comumente utilizado em simulações de redes móveis. No entanto, o RWP abstrai o padrão de mobilidade humana. Os nós se movem com velocidade entre 0.5m/s e 1.5m/s.
- **Small World in Motion** (SWIM), explora a regularidade da mobilidade humana e gera uma mobilidade sintética com padrões similares a diversos registros de mobilidade reais (*traces*) disponíveis na literatura. Mei *et al*. mostraram que os protocolos de encaminhamento oportunísticos apresentam resultados semelhantes quando analisados utilizando o SWIM ou registros de mobilidade humana reais. Os parâmetros do SWIM foram definidos com os mesmos valores utilizados em [Mei and Stefa 2009].

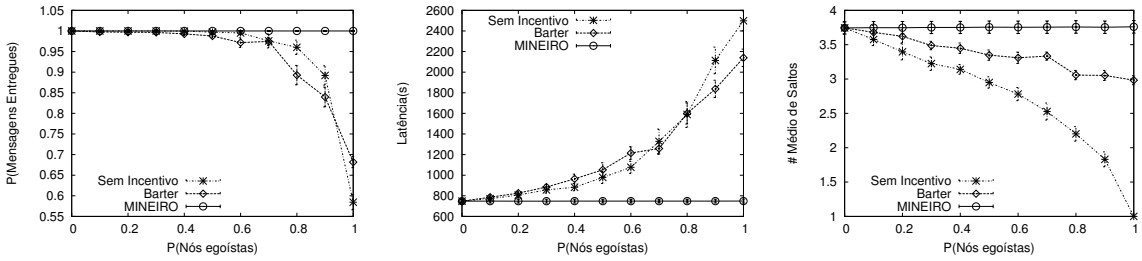
Por uma questão de simplicidade, todos os nós têm capacidade de armazenamento no *buffer* infinita, Δ_{inc} e Δ_{dec} são iguais a 0.1 e I é igual a 0.5, e estes valores seguem a equação derivada para o equilíbrio bayesiano.

Todos os resultados apresentam a média e o intervalo de confiança de 95% das quinze execuções para cada cenário. Analisamos as seguintes métricas para avaliar o desempenho da rede em relação à percentagem de nós egoístas na rede.

- **Taxa de Entrega de Mensagens**: A quantidade de mensagens enviadas que atingiram o destino dividido pelo número de mensagens criadas.
- **Latência Média**: O tempo entre a criação de uma mensagem até o seu destino.
- **Média de Saltos por mensagem**: Número médio de nós que retransmitiram uma mensagem da origem até seu destino.

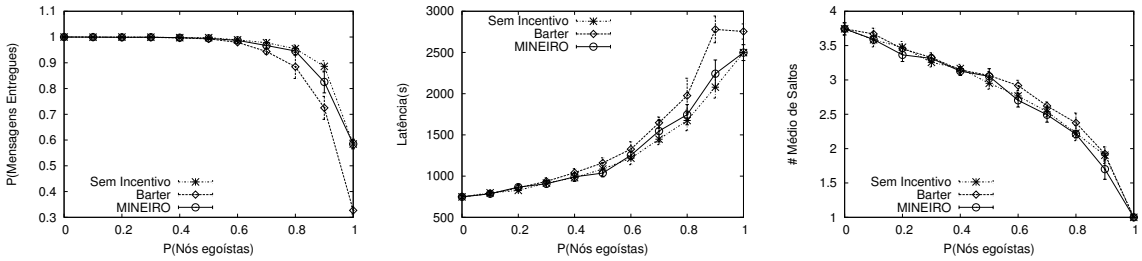
5.2. Impacto do Comportamento Egoísta nas Redes Oportunísticas

A percentagem de nós egoístas na rede, $\mathcal{R}_v = 0$, variou de 0 a 100%. Consideramos dois tipos de egoísmo: i) *não-encaminhamento*: nós mantém a mensagem em seu *buffer* sem encaminhá-la; ii) *descarte de mensagem*: nós descartam a mensagem recebida e todas as informações sobre ela.



(a) Taxa de Entrega **(b)** Latência **(c)** Saltos por Mensagem

Figura 2. Comportamento egoísta de não-encaminhamento de mensagens no modelo de mobilidade RWP.



(a) Taxa de Entrega **(b)** Latência **(c)** Saltos por Mensagem

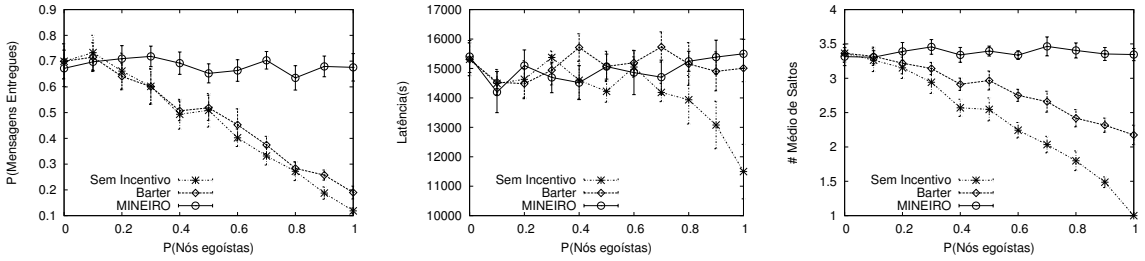
Figura 3. Comportamento egoísta de descarte de mensagens no modelo de mobilidade RWP.

Estes dois tipos de egoísmo diferem-se pelo impacto causado nas retransmissões de mensagens. Quando dois nós se encontram, eles trocam uma lista com os identificadores das mensagens que possuem em seus *buffers*, verificam as mensagens em comum, e encaminham apenas as mensagens que o outro nó não possui[2]. O comportamento de *não-encaminhamento* evita que o nó egoísta receba uma mensagem que já possui. Já no *descarte de mensagem*, as retransmissões de uma mesma mensagem para o mesmo nó aumentam, pois um nó descarta uma mensagem e pode recebê-la em outras ocasiões.

Nas Figuras 2 e 3 são mostrados os resultados para o cenário utilizando o modelo de mobilidade RWP. Neste cenário, a taxa de entrega mantém-se constante até a presença de 60% de nós egoístas, Figura 2a. A entrega nos mecanismos sem incentivo e com incentivo por permutação de mensagens (*barter*) decresce rapidamente após este valor. Além disso, a latência aumenta linearmente após 60% (Figura 2b), enquanto o mecanismo MINEIRO mantém-se constante para estas métricas. Isto indica que neste cenário a rede tolera até 60% de nós com comportamento egoísta sem queda de desempenho.

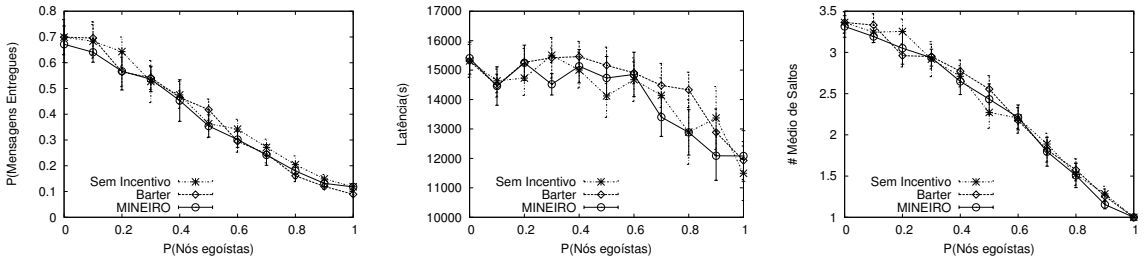
A Figura 2c explica este resultado: com o aumento de nós egoístas, o número de saltos médios diminui. No pior caso, quando todos os nós se tornaram egoístas, o número de saltos médio foi igual a um (Sem Incentivo), o que significa que as mensagens foram entregues diretamente ao destino. No MINEIRO e no *Barter* os nós são racionais, ou seja, um nó deve repassar mensagens para que possa transmitir suas próprias mensagens. No MINEIRO, os nós egoístas recebem um crédito inicial, o que incentiva os nós a trocarem mensagens de terceiros, mantendo assim o número de saltos constante.

[2]A identificação das mensagens que cada nó possui em seu *buffer* antes de transmití-las é comum na maioria dos protocolos de contato em redes oportunísticas, pois evitam retransmissões desnecessárias.



(a) Taxa de Entrega **(b)** Latência **(c)** Saltos por Mensagem

Figura 4. Comportamento egoísta de não-encaminhamento de mensagens no modelo de mobilidade SWIM.



(a) Taxa de Entrega **(b)** Latência **(c)** Saltos por Mensagem

Figura 5. Comportamento egoísta de descarte de mensagens no modelo de mobilidade SWIM.

Na Figura 3 são apresentados os resultados quando os nós egoístas descartam as mensagens. Neste caso, a rede tem uma tolerância de até 60% de egoísmo. No entanto, os mecanismos de incentivo não foram capazes de manter o desempenho da rede.

Os resultados para o modelo de mobilidade SWIM são apresentados nas Figuras 4 e 5. A taxa de entrega é menor que a obtida no cenário utilizando o RWP devido à criação de comunidades. Mensagens geradas para nós externos à comunidade da fonte têm probabilidade menor de alcançar o destino. A taxa de entrega para o MINEIRO permanece constante quando o comportamento egoísta é de *não-encaminhamento* (Figura 4). Novamente, o MINEIRO mantém o número de saltos constante, estimulando os nós a encaminharem as mensagens de terceiros. O comportamento egoísta de *descarte de mensagens*, Figura 5, causa uma degradação quase linear em todas as métricas para todos os mecanismos analisados. Essa degradação é causada principalmente pelo fato de que as mensagens foram geradas entre pares aleatórios de origem e destino. Se o destino da mensagem não pertence ao grupo da origem, as chances de uma mensagem ser entregue são menores. No entanto, o MINEIRO manteve um comportamento quase linear quando o comportamento dos nós egoístas foi de não encaminhar.

Entre todos os cenários analisados, como mostrado nas Figuras 3 e 5, o comportamento egoísta com a política de *descarte de mensagens* apresentou os piores resultados em termos de desempenho da rede. De fato, quando os nós egoístas descartam as mensagens recebidas, o MINEIRO tem resultados semelhantes aos demais. A política de descarte de mensagens causa um aumento no número de retransmissões de uma mesma mensagem para um mesmo nó intermediário, pois o nó pode solicitar uma mensagem que já descartou diversas vezes.

6. Conclusão

Neste trabalho propusemos MINEIRO, um mecanismo de incentivo que atribui reputação a nós com base na origem das mensagens que os mesmos enviaram. O MINEIRO dispensa o requisito de uma entidade central para garantir informações sobre reputação dos nós, tal como é feito em outros mecanismos.

Os resultados das simulações mostraram que, em uma rede com mobilidade aleatória, a taxa de entrega e o atraso não sofrem alterações se até 60% dos participantes da rede tiverem comportamento egoísta. Após este valor, somente o MINEIRO manteve a taxa de entrega e o atraso constante. Em redes baseadas em comunidades sociais, o comportamento egoísta degrada o desempenho da rede rapidamente. Nossos estudos também mostraram que descartar as mensagens degrada a taxa de entrega mais rapidamente do que quando estas são mantidas no *buffer* mas não são encaminhadas.

Há diversas questões de segurança associadas às redes oportunísticas que estão fora do escopo deste trabalho, e as deixamos como trabalhos futuros. Como exemplo, citamos a falsificação da origem da mensagem, que poderia levar o MINEIRO a calcular erroneamente a reputação. Uma solução para garantir a autenticidade das mensagens é certificá-las por meio de mecanismos de chaves públicas e privadas, o que exigiria uma entidade central para criação e gerenciamento destas chaves. Contudo, assumimos que redes oportunísticas formadas por dispositivos móveis são redes abertas, nas quais novos participantes podem entrar a qualquer momento. Neste caso, manter uma entidade central para gerenciar estas chaves torna-se um desafio.

Agradecimentos

Os autores gostariam de agradecer o suporte da Capes, CNPq e Fapemig.

Referências

(Updated September 2014). Wi-fi certified wi-fi direct: Personal, portable wi-fi to connect devices anywhere, anytime. *WiFi Alliance*.

3GPPP (2013). Study on Architecture Enhancements to Support Proximity Services (ProSe) (Release 12). Report 23.703 V0.4.1, 3rd generation partnership project.

Adar, E. and Huberman, B. A. (2000). Free riding on gnutella. *First Monday*, 5(10).

Bigwood, G. and Henderson, T. (2011). Ironman: Using social networks to add incentives and reputation to opportunistic networks. In *Privacy, security, risk and trust Workshop in IEEE Socialcom*, pages 65–72. IEEE.

Bowles, S. and Gintis, H. (2003). *Origins of human cooperation. In: Genetic and cultural evolution of cooperation*. MIT Press Cambridge, MA.

Buttyán, L., Dóra, L., Félegyházi, M., and Vajda, I. (2010). Barter trade improves message delivery in opportunistic networks. *Ad Hoc Networks*, 8(1):1–14.

Crowcroft, J., Gibbens, R., Kelly, F., and Östring, S. (2004). Modelling incentives for collaboration in mobile ad hoc networks. *Performance Evaluation*, 57(4):427–439.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. *Applications, technologies, architectures, and protocols for computer communications*, pages 27–34.

Gaoning, H., Debbah, M., and Lasaulce, S. (2010). *Game Theory for OFDM Systems with Incomplete Information. In: Game Theory for Wireless Communications and Networking*. CRC Press.

Hui, P., Xu, K., Li, V., Crowcroft, J., Latora, V., and Lio', P. (2009). Selfishness, altruism and message spreading in mobile social networks. In *INFOCOM Workshops 2009, IEEE*, pages 1–6.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The ONE Simulator for DTN Protocol Evaluation. In *2nd International Conference on Simulation Tools and Techniques*.

Lee, J. and Hoh, B. (2010). Sell your experiences: a market mechanism based incentive for participatory sensing. In *PerCom*, pages 60–68. IEEE.

Levine, D. K. (1998). Modeling altruism and spitefulness in experiments. *Review of economic dynamics*, 1(3):593–622.

Mei, A. and Stefa, J. (2009). Swim: A simple model to generate small mobile worlds. In *IEEE INFOCOM*, pages 2106–2113. IEEE.

Miao, J., Hasan, O., Mokhtar, S. B., Brunie, L., and Yim, K. (2013). An investigation on the unwillingness of nodes to participate in mobile delay tolerant network routing. *International Journal of Information Management*, 33(2):252–262.

Mota, V. F. S., Cunha, F. D., Macedo, D. F., Nogueira, J. M. S., and Loureiro, A. A. F. (2014). Protocols, mobility models and tools in opportunistic networks: A survey. *Computer Communications*.

Panagakis, A., Vaios, A., and Stavrakakis, L. (2007). On the Effects of Cooperation in DTNs. *2nd International Conference on Communication Systems Software and Middleware*, pages 1–6.

Vahdat, A. and Becker, D. (2000). Epidemic routing for partially connected ad hoc networks. *Tec. Report, Duke University*.

Wei, L., Cao, Z., and Zhu, H. (2011). MobiGame: A User-Centric Reputation Based Incentive Protocol for Delay/Disruption Tolerant Networks. *Communications Society*.

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.

Yang, D., Xue, G., Fang, X., and Tang, J. (2012). Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *Mobicom*, pages 173–184.

Estratégias para Resiliência em SDN : Uma Abordagem Centrada em Multi-Controladores Ativamente Replicados

Eros S. Spalla[1], Diego R. Mafioletti[1], Alextian B. Liberato[1]
Christian Rothenberg[2], Lasaro Camargos[3], Rodolfo da S. Villaca[1], Magnos Martinello[1]

[1]Universidade Federal do Espirito Santo (UFES) – Vitória/ES
[2]Universidade Estadual de Campinas (UNICAMP) – Campinas/SP
[3]Universidade Federal de Uberlândia (UFU) – Uberlândia/MG

spalla@ifes.edu.br, diego@saorc.com.br, alextian@ifes.edu.br
chesteve@dca.fee.unicamp.br, lasaro@ufu.br
rodolfo.villaca@ufes.br, magnos@inf.ufes.br

Abstract. *Software Defined Networking (SDN) are based on the separation of control and data planes. The SDN controller, although logically centralized, should be effectively distributed for high availability. Since the specification of OpenFlow 1.2, there are new features that allow the switches to communicate with multiple controllers that can play different roles –* master*,* slave*, and* equal*. However, these roles alone are not sufficient to guarantee a resilient control plane and the actual implementation remains an open challenge for SDN designers. In this paper, we explore the OpenFlow roles for the design of resilient SDN architectures relying on multi-controllers. As a proof of concept, a strategy of active replication was implemented in the Ryu controller, using the OpenReplica service to ensure consistent state among the distributed controllers. The prototype was tested with commodity RouterBoards/MikroTik switches and evaluated for latency in failure recovery and switch migration for different workloads. We observe a set of trade-offs in real experiments with varyin workloads at both the data and control plane.*

Resumo. *As Redes Definidas por Software (SDN) separam os planos de dados e de controle. Embora o controlador seja logicamente centralizado, ele deve ser efetivamente distribuído para garantir alta disponibilidade. Desde a especificação OpenFlow 1.2, há novas funcionalidades que permitem aos elementos da rede se comunicarem com múltiplos controladores, que podem assumir diferentes papéis –* master*,* slave*, e* equal*. Entretanto, esses papéis não são suficientes para garantir resiliência no plano de controle, pois delega-se aos projetistas de redes SDN a responsabilidade por essa implementação. Neste artigo, exploramos os papéis definidos no protocolo OpenFlow no projeto de arquiteturas resilientes SDN com base em multi-controladores. Como prova de conceito uma estratégia de replicação ativa foi implementada no controlador Ryu usando o serviço OpenReplica para garantir a consistência dos estados. O protótipo foi testado com* switches *RouterBoards/MikroTik comerciais avaliando-se a latência na recuperação de falha e na migração de switches entre controladores. Observamos diferentes compromissos de projeto em experimentos reais sujeitos a várias cargas nos planos de dados e de controle.*

Agradecimentos à CAPES, CNPq e FAPEMIG pelo suporte à execução deste trabalho.

1. Introdução

A flexibilidade oferecida pela arquitetura das Redes Definidas por *Software* (SDN – *Software-Defined Networking*) apoia-se na separação do plano de controle do plano de dados, permitindo a implementação das funções de controle de rede a partir de uma visão centralizada [Kreutz et al. 2015]. Nesta abordagem, o estado dos dispositivos da rede é determinado por um controlador, que envia suas decisões ao dispositivo, sob a forma de entradas nas tabelas de fluxos, por meio do protocolo OpenFlow [Rothenberg et al. 2010].

Atualmente, o protocolo OpenFlow é considerado o padrão *de facto* em SDN. Porém, garantir alta disponibilidade no plano de controle é um desafio em aberto para o qual o protocolo OpenFlow não oferece uma solução pronta, e sim um conjunto de possibilidades. Desde a versão 1.2 da especificação, o OpenFlow tem incorporado novas funcionalidades que permitem aos *switches* se comunicarem com múltiplos controladores. Uma dessas funcionalidades são os papéis (*roles*) de *master*, *equal* e *slave*, assumidos por múltiplos controladores na comunicação com os dispositivos no plano de dados. Entretanto, apenas essas novas *features* não são suficientes para garantir uma solução resiliente, uma vez que, o padrão OpenFlow define os papéis, mas deixa a cargo dos projetistas de redes SDN a escolha de estratégias para implementá-las.

Uma onda de trabalhos recentes tem abordado o problema de resiliência em redes SDN (ex: Sec.V [Kreutz et al. 2015]), incluindo o posicionamento de controladores distribuídos [Heller et al. 2012], clustering [Penna et al. 2014], particionamento do controle [Koponen et al. 2010], garantias de consistência [Botelho et al. 2013], entre outros. Uma característica comum a todos os trabalhos relacionados que encontramos é não terem explorado as opções de conectividade simultânea com múltiplos controladores, usando as três opções de papéis disponíveis no protocolo OpenFlow.

A primeira contribuição do trabalho, é apresentar e discutir diferentes estratégias de replicação do plano de controle e seus *trade-offs*, abordando aspectos práticos de cada uma. Nossa proposta baseia-se no levantamento das características e componentes que tornam esses sistemas funcionais, tais como a relação dos papéis OpenFlow dos controladores e a distribuição de estados, assim como o comportamento da rede em caso de falha e restauração nos controladores.

Como prova de conceito de um plano de controle resiliente, uma estratégia de replicação ativa foi implementada no controlador Ryu[1], e o protótipo foi testado com *switches* RouterBoards comerciais, modificados por uma arquitetura aberta [Liberato et al. 2014], onde foram observados a latência na recuperação de falha e restauração de falha/migração de *switches* entre controladores. A descrição do protótipo implementado e a análise experimental são nossa segunda contribuição ao estado da arte, já que trabalhos relacionados apenas consideraram ambientes emulados.

O restante deste artigo está estruturado da seguinte forma: a Seção 2 discute diferentes abordagens para implementação de resiliência em controladores OpenFlow; a Seção 3 apresenta uma proposta para controladores resilientes; a Seção 4 indica a metodologia experimental; a Seção 5 expõe os resultados obtidos e aponta o comportamento do experimento em um ambiente real; a Seção 6 reporta os trabalhos relacionados; e por fim, a Seção 7 tece considerações finais e indica os trabalhos futuros.

[1] https://github.com/osrg/ryu

2. O problema arquitetura multi-controlador

As características arquiteturais e inovações advindas das SDN são objeto de inúmeros estudos e discussões [Kreutz et al. 2015]. Nesse contexto, um ponto controverso é justamente uma de suas principais características: a centralização do controle. Isto porque, caso tal visão centralizada seja implementada por um único controlador, uma falha nesse elemento pode levar à indisponibilidade total da rede. O problema fundamental e os *trade-offs* por trás da distribuição do controle em redes SDN podem ser modelados adaptando o teorema de CAP [Panda et al. 2013]. [2]

Embora tenha ficado claro desde a especificação inicial do OpenFlow que o controlador, apesar de logicamente centralizado, poderia ser implementado de forma distribuída, somente a partir da versão 1.2 é que os mecanismos para esta distribuição começaram a ser especificados. Nas versões 1.0 e 1.1, cada *switch* se conecta apenas a um controlador, que torna-se um ponto único de falha e demanda ações não especificadas para reposição. Desde a versão 1.2, *switches* OpenFlow podem se conectar a diversos controladores simultaneamente, que devem assumir um dentre três papéis: *master*, *slave*, e *equal*. Contudo, na literatura, não há sequer *guidelines* ou melhores práticas de como estes papéis devam ser usados. Assim, cabe aos projetistas de SDN o uso de técnicas que garantam a disponibilidade dos serviços executados pelo controlador e a dependabilidade da rede. A abordagem óbvia para se atender tal requisito é a redundância do estado do controlador via alguma forma de replicação, como até é sugerido pelos nomes dos papéis, permitindo que outro controlador continue a atuar sobre o mesmo estado, em caso de falha do primeiro ou, ainda, que múltiplos controladores compartilhem tal tarefa.

Na implementação da replicação em ambiente SDN multicontrolador, diversas perguntas precisam respostas, dentre elas destacamos as seguintes: Vários controladores processam comandos que atualizam o estado ou somente um processa e informa aos outros o novo estado? No primeiro caso, a mesma ordem de processamento é imposta para garantir consistência e, no segundo, como o estado é compartilhado? Todos acessam o estado global ou ele é particionado entre os controladores? Como são mascaradas as falhas de um coordenador? Na seção seguinte, discutiremos tais perguntas e possíveis respostas.

2.1. Suporte a múltiplos controladores

Em uma rede implementada com o protocolo OpenFlow, os *switches* mantém tabelas de fluxo que ditam como o tráfego deve ser roteado. Quando um pacote de dados não pode ser roteado, o controlador responsável pelo *switch* é informado via um evento *packet-in*. Baseado em seu estado, o controlador gera uma nova regra a ser inserida na tabela de fluxo do *switch* para lidar com o pacote e outros similares. Assim, se há apenas um controlador por *switch*, a tabela de fluxos é sempre consistente com o estado do controlador.

Contudo, a partir da definição do conceito de papéis, um *switch* pode se conectar a mais de um controlador. Neste caso, quando a conexão ocorre, por padrão, o controlador assume o papel *equal*, implicando que todos os controladores têm igual poder de atualizar tabelas do *switch*. Neste cenário os controladores devem, de uma forma não especificada,

[2] O teorema CAP (Consistency, Availability e Partition Tolerance), também cohecido como Teorema de Brewer, afirma afirma a impossibilidadem de um sistema computacional distribuído garantir simultaneamente consistência, disponibilidade e tolerância ao particionamento.

coordenar a atualização de seus estados, mantendo uma visão logicamente centralizada da rede e garantindo que suas intervenções sejam consistentes.

Outros papéis que podem ser assumidos pelo controlador são *slave* e *master*. Ao assumir o papel *master*, assim como no *equal*, o controlador passa a ter privilégios sobre o *switch*, com permissão de leitura e escrita, além de receber mensagens assíncronas. Já no modo *slave*, o controlador tem limitado seu acesso ao *switch*, podendo realizar apenas operações de leitura, isto é, nenhuma mensagem que altere a tabela de fluxos do *switch* é processada. Caso comandos de escrita sejam enviados ao *switch*, uma mensagem de erro será gerada. Mensagens assíncronas também não são enviadas ao controlador, à exceção das mensagens de *port-status*, que informam, por exemplo, sobre o estado das portas.

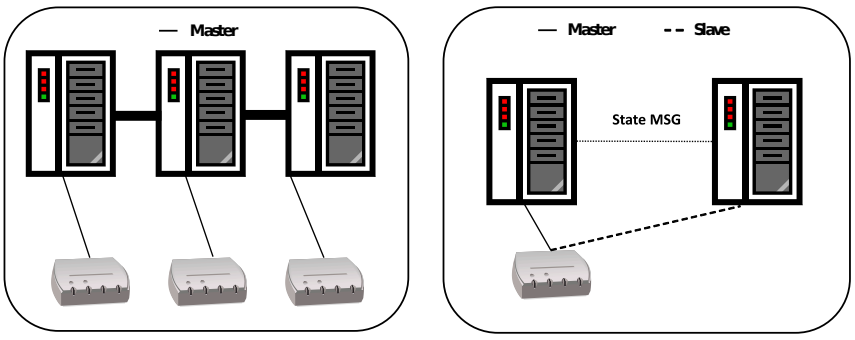
2.2. Configuração Equal

Com base nas definições dos papéis OpenFlow, podemos descrever uma configuração básica para múltiplos controladores rodando uma aplicação de *learning switch*:

1. um *switch* se conecta a *N* controladores;
2. todos os controladores assumem o papel *equal*.

Neste arranjo simples, todos os *switches* são configurados com os endereços dos controladores, e estes não demandam nenhuma configuração, uma vez que o papel padrão no OpenFlow é o *equal*. Um *switch* ao receber um pacote que não tenha fluxo instalado em sua tabela, encaminha um *packet-in* ao plano de controle, isto é, a todos os controladores. Tais controladores, ao receberem o pacote, realizam o seu processamento de forma concorrente, e enviam um *packet-out* para o *switch*. Desde que processem os mesmos pacotes, na mesma ordem, e que o processamento seja determinístico, os estados dos controladores evoluirão consistentemente. Neste cenário, em caso de falha, os controladores restantes continuam a operar a rede, normalmente. Este é o princípio da replicação de máquinas de estados [Lamport 1978, Schneider 1990], também conhecido como *Replicação Ativa*, ilustrado na Figura 1(a). Obviamente, a dificuldade na implementação desta técnica está em garantir a entrega, totalmente ordenada, das mensagens aos controladores, o que requer alguma forma de comunicação em grupo [Défago et al. 2004].

Apesar de conceitualmente simples, essa implementação tem desvantagens claras: maior uso de recursos – já que todos os controladores processam todos os pacotes;



(a) Replicação Ativa (b) Replicação Passiva

Figura 1. Arquiteturas Multicontroladores

sobrecarga da rede de controle – devido ao *broadcast* dos *packet-in* e múltiplas respostas; a capacidade de processamento do plano de controle não escala com o número de controladores, já que todos processam os mesmos pacotes e mantém o mesmo estado; e por fim, a menos que os comandos que alteram as tabelas de fluxo sejam idempotentes, o resultado pode não refletir o que foi determinado pelos controladores – por exemplo, em nossos experimentos, observamos a duplicação de pacotes causada pelo processamento de comandos duplicados pelo *switch*.

2.3. Configuração Master-Slave

Uma solução alternativa, que minimiza alguns dos efeitos indesejáveis listados anteriormente, consiste em ter apenas um dos controladores processando as mensagens e respondendo ao *switch*, e repassando suas mudanças de estado aos outros controladores. Esta técnica, conhecida como *Replicação Passiva*, *primary-backup* ou, finalmente, *master-slave*, ilustrado na Figura 1(b) [Budhiraja et al. 1993] diminui o processamento nos controladores, elimina o não determinismo na atualização do estado e evita as respostas redundantes e seus efeitos. Do ponto de vista do *OpenFlow*, esta técnica pode ser implementada da seguinte forma:

1. um controlador assume o papel *master* para todos os *switches* da rede;
2. os demais controladores assumem o papel *slave* para todos os *switches* da rede.

Assim, como no caso da replicação ativa, apenas atribuir os papéis não é o suficiente para levar a uma solução completa. Se na replicação ativa era necessário garantir a entrega e ordenação das mensagens, na passiva é necessário garantir que as atualizações feitas no controlador mestre sejam replicadas para os escravos, novamente usando alguma forma de comunicação em grupo. Além disso, é também necessário que os escravos monitorem o mestre e que um deles assuma seu papel no caso de falha, informando os *switches*.

2.4. Configuração Multi-Master / Multi-Slave

A terceira proposta discutida para prover resiliência ao plano de controle, utiliza mecanismos para que os controladores não sejam o gargalo da rede. Em um arranjo com M *switches* e N controladores pode-se ter a seguinte configuração:

1. para cada *switch* da rede, um controlador deve assumir o papel *master*;
2. para cada *switch* da rede, $N - 1$ controladores devem assumir o papel *slave*.

Essa proposta permite que os N controladores processem *packet-in*, já que os *switches* são organizados de modo que um controlador atue como *master* para um *switch* e *slave* para os demais. Nos *switches* nenhuma configuração extra é necessária, além dos endereços dos controladores, levando a uma rede operacional com "mínimo" esforço. Todavia, ainda existem questões que precisam ser respondidas, sendo provavelmente a mais importante a seguinte: se os estados dos *switches* estão particionados entre os controladores, como implementar a visão centralizada do controlador OpenFlow?

2.5. Configuração via Data Store

Uma variação das configurações anteriores é utilizar um *data store* externo, tolerante a falhas, no qual o mestre espelha seu estado interno, e que é usado pelos escravos para atualizarem seus estados. Com o uso desta abstração extra, os controladores a utilizam para manter atualizados os seus estados; a visão centralizada passa a ser implementada em uma

base externa, e a visão particionada, por exemplo, corresponderia a simplesmente o controlador ignorar parte dos dados da base externa. Tal arquitetura tem a vantagem de abstrair a comunicação entre controladores, possivelmente simplificando a implementação.

Outra questão importante é como os papéis são divididos entre os *switches*. Novamente um serviço externo pode ser usado, como por exemplo os serviços de coordenação ZooKeeper [Hunt et al. 2010] ou Doozer[3]. Esses serviços podem ser usados para monitorar as réplicas e, em caso de falhas, eleger um mestre para cada *switch*. O novo mestre deve então agregar o estado do controlador que falhou ao seu, a partir das informações contidas no *data store*, e, por fim, informar os *switches* "órfãos" quem é seu novo *master*.

2.6. Considerações parciais

Algumas considerações podem ser feitas em relação a essa última perspectiva de um plano de controle resiliente. O processamento de pacotes por múltiplos controladores de forma concorrente é uma interessante escolha, visto que, uma importante característica para sistemas distribuídos é adicionada: escalabilidade. A medida que controladores são adicionados a rede, o *throughput* do plano de controle aumenta. No entanto, ter o estado da rede distribuído entre os controladores impõe desafios aos projetistas de redes SDN, tais como: a manutenção da consistência das informações e a coordenação das réplicas. Para avaliar essas conjecturas e os *trade-offs* em implementações reais, e avançar na viabilidade de uma solução de SDN com alta disponibilidade, optamos por desenvolver uma prova de conceito.

3. Proposta de Arquitetura Resiliente e Implementação do Protótipo

Após as discussões sobre diferentes estratégias e relações de compromisso na implementação de redes SDN com múltiplos controladores, nesta seção é apresentada a proposta de um plano de controle resiliente e o protótipo implementado.

Com o objetivo de lidar com os diferentes *trade-offs* apresentados nos exemplos anteriores, os seguintes pré-requisitos foram priorizados: (1) todos os controladores devem estar aptos a processar *packet-in*, isto é, atuar com o papel *master* de forma concorrente; (2) a distribuição de estados deve ser ativa, levando à resiliência no plano de controle, enquanto o estado da rede é replicado proativamente entre os controladores.

O protótipo implementado faz uso de dois controladores e duas RouterBoards RB2011iLS-IN, organizados de acordo com a Figura 2(a).

Um importante ponto no projeto de um plano de controle resiliente, é a implementação da visão centralizada (estado da rede compartilhado de forma consistente) em controladores distribuídos. Nossa proposta utiliza o OpenReplica[4] para manter a consistência do estado da rede. OpenReplica é um serviço que provê replicação e sincronização para sistemas distribuídos em larga escala. Baseado em uma nova implementação de máquinas de estados replicadas via Paxos [Lamport 1998], que utiliza uma abordagem orientada a objetos, o sistema, ativamente, cria e mantém réplicas vivas para objetos fornecidos pelo usuário. Assim, de forma transparente aos clientes, os objetos replicados são acessados como se fossem objetos locais (vide Figura 3).

[3]https://github.com/ha/doozerd

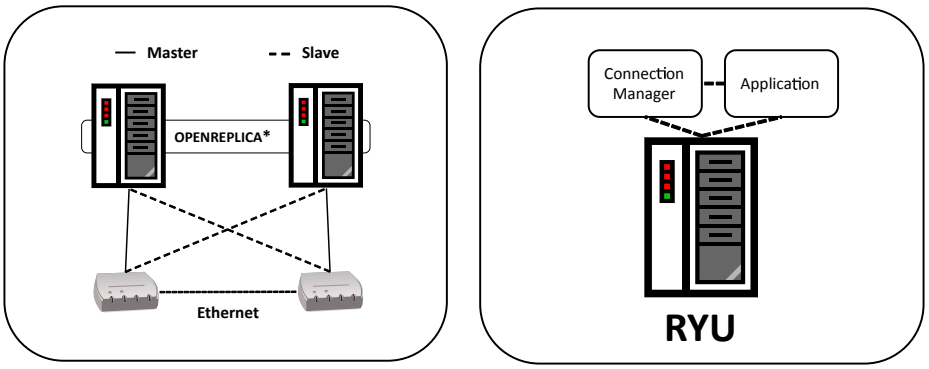
[4]http://openreplica.org/

A Figura 2(b) apresenta a organização do controlador com as funções do sistema divididas em dois módulos. O primeiro, chamado *ConnectionManager*, gerencia a comunicação entre os controladores, coordena as operações de troca de papéis, recuperação e restauração de falhas; já o segundo, chamado *Application*, carrega a aplicação a ser executada, neste caso, um *learning switch* com suporte ao OpenFlow 1.3.

Ao iniciar a execução dos controladores, os módulos são carregados em sequência, sendo *ConnectionManager* o primeiro. Como a conexão de todos *switches* da rede aos controladores é uma premissa para o funcionamento do sistema, o *ConnectionManager* mantém uma função aguardando os eventos de conexão dos *switches*. À medida que *switches* conectam-se aos controladores, o módulo *ConnectionManager* executa uma operação que configura os papéis para cada *switch*, na qual cada *switch* se conecta a um controlador com o papel *master* e a *N* controladores com o papel *slave*. De modo a tornar essa operação mais simples, já que o evento de conexão dos *switches* é aleatório, os controladores ordenam esses eventos de acordo com o *dpid* do *switch*. Por exemplo, suponha dois *switches*: 01 e 02. Estes podem conectar-se aos controladores em qualquer ordem, entretanto, os controladores ao ordenarem esses eventos, garantem que a conexão sempre será feita primeiro pelo *switch* 01 e depois pelo *switch* 02.

Os controladores são responsáveis pela detecção das falhas do plano de controle, e para tratar estes eventos, o protótipo utiliza dois procedimentos: recuperação de falha, e a restauração de falha. A situação de falha ocorre quando há indisponibilidade de um dos controladores. Na implementação protótipo, é considerado um modelo de sistema assíncrono com falhas de controladores, do tipo *fail-stop*; os canais de comunicação não duplicam nem perdem mensagens, e a perda de conexão é utilizada como indicador de falha nos controladores

A **recuperação de falha** consiste em um processo que faz com que o controlador, que continua ativo, torne-se o novo *master* para todos os *switches* que perderam o seu controlador principal. Os passos executados na operação, são: no momento em que a falha é detectada, o módulo *ConnectionManager* executa uma função que verifica quais são os *switches* que têm como *master* o controlador que falhou, e os envia uma mensagem *role-request master*. Cada *switch* responde com uma mensagem *role-reply* para confirmar



(a) Proposta de Resiliência para o plano de controle OpenFlow

(b) Organização do Controlador

Figura 2. Protótipo Implementado.

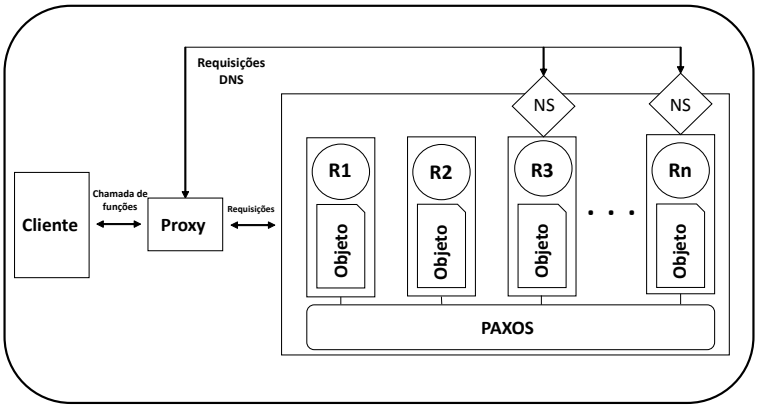


Figura 3. OpenReplica - Replicação e sincronização do estado da rede.

a operação. Já a restauração, ocorre quando o controlador que falhou se recupera da falha, ou seja, está apto novamente a processar pacotes. Nesse caso, o controlador, que volta da falha, executa o protocolo de migração de *switches* e recupera os *switches* que o tinham como *master* no momento anterior a falha.

No processo de **restauração de falha**, nossa proposta apresenta uma pequena variação da técnica para migração de *switches* entre controladores em [Dixit et al. 2013]. O controlador que volta de uma falha, no momento da inicialização de seus módulos, já carrega os *switches* em modo *equal*. Esse procedimento otimiza a execução da migração, já que as mensagens de troca de papel são enviadas na inicialização do controlador. Após essa operação, uma mensagem de migração é enviada para o controlador que já estava ativo, e então, a mesma sequência descrita por [Dixit et al. 2013] é seguida.

4. Avaliação Experimental

Na avaliação experimental foram usados *switches* comerciais modificados conforme [Liberato et al. 2014]. A ideia é explorar a capacidade de alguns equipamentos existentes no mercado, que podem ser usados para prototipação, com desempenho compatível e baixo custo. Com base em experimentos realizados com RouterBoards, é possível inferir como o protótipo se comporta em um ambiente real.

O ambiente utilizado para realizar os experimentos é um computador (Core i5 3.2GHz, 6GB de RAM, HD 320GB 5400rpm, SO Ubuntu 14.04 64bits), executando os controladores; duas Routerboards RB2011iLS-IN; e dois computadores (CPU AMD Athlon 64 X2 4000+, 2GB de RAM, HD 160GB 7200rpm, SO Ubuntu 14.04 64bits).

4.1. Metodologia

Para permitir o mínimo de interferência entre os cenários, os equipamentos foram reinicializados em todos os experimentos, garantindo que informações contidas no *cache* não influenciassem os resultados. Nos testes e na avaliação da proposta apresentada os seguintes parâmetros foram medidos: latência de recuperação e restauração de falha dos controladores, com diferentes taxas de tráfego no plano de controle; latência de *packet-in* nos controladores, antes e após a falha; latência de recuperação e restauração de falha dos controladores, com diferentes taxas de tráfego no plano de dados; e o consumo de processador das Routerboards nos diferentes cenários apresentados.

4.2. Carga de trabalho

De modo a gerar o tráfego necessário de acordo com as demandas de cada cenário, foi criado um gerador de tráfego em Perl que utiliza a extensão NET::RawIP para criar pacotes IP a uma taxa pré-definida. No ambiente de experimentação, os controladores não foram o *bottleneck* dos testes, ao contrário, por conta do *hardware* utilizado, seria possível aumentar a taxa de envio de pacotes. Devida a limitação do *hardware* utilizado, especial da *CPU*, limitamos a taxa de geração de pacotes a no máximo 1000 *packet-in/s*. Então, as taxas de envio pré-definidas foram: 250, 500, 750 e 1.000 *packet-in/s*.

Para os cenários com propósito de gerar tráfego no plano de controle, a aplicação de *switching* foi alterada para que todos os pacotes gerados pelos *hosts* fossem enviados aos controladores, ou seja, não haviam regras instaladas nos *switches*. Já para os testes com foco no plano de dados, foram instaladas regras apenas para fazer o encaminhamento entre os dois *hosts* da rede, assim, nenhum *packet-in* foi enviado aos controladores.

Em cenários cujo objetivo era gerar tráfego no plano de dados, utilizou-se a ferramenta Iperf [5]. Com o *throughput* das RouterBoards conhecido, as taxas de envio foram definidas de forma que houvesse um aumento gradual até o limite dos equipamentos. As taxas pré-definas foram: 180, 360, 540 e 720 Mbps.

4.3. Métricas

Para cada parâmetro a ser medido coletaram-se 30 amostras nos diferentes cenários. Foi calculado o intervalo de confiança de 95% para essas amostras e este se mostrou aceitável para garantir a confiabilidade do resultado as experimentações.

Para medir as operações realizadas pelos controladores, foram inseridos quatro *timestamps* no código. Na recuperacao de falha, assim que o controlador envia a mensagem *change-role-request* ao *switch*, e logo após a recepção da mensagem *change-role-reply*; na restauração de falha, logo que a migração é iniciada, e no momento em que o controlador, que voltou da falha, recebe a mensagem de término do processo.

Na medição da latência dos pacotes nos controladores, o tráfego foi capturado por meio da ferramenta *tshark*, e posteriormente analisada com o *wireshark*[6], de modo que cada *packet-in* tivesse o tempo medido de acordo com o seu *packet-out*. Foram coletados pacotes antes e após as simulações de falha.

Finalmente, para medir o percentual de uso de CPU das RouterBoards, a ferramenta de monitoramento *mpstat*[7] foi utilizada.

5. Discussão dos Resultados

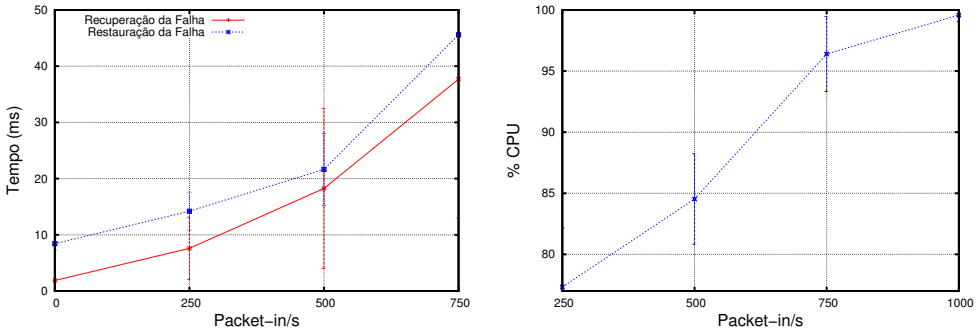
Nesta seção discute-se os resultados obtidos, organizados na seguinte sequência: primeiro, são apresentados os resultados dos testes com foco no plano de controle; em seguida, os números do plano de dados; e por fim, uma análise dos resultados.

Os valores de 1.000 *packet-in/s* e 720 *Mbps* representam o limite prático para o protótipo, isto é, não recomenda-se a utilização dos equipamentos com essa carga de

[5]https://github.com/esnet/iperf

[6]https://www.wireshark.org/

[7]http://www.linuxcommand.org/man_pages/mpstat1.html



(a) Recuperação e Restauração da Falha (b) Carga de Processamento nas RouterBoards

Figura 4. Tráfego no Plano de Controle

trabalho. Assim, de modo a melhorar a visualização dos gráficos, a latência das operações de Recuperação e Restauração da Falha, com taxas de 1.000 *packet-in/s* e 720 *Mbps*, não são apresentadas nos gráficos, apenas no texto.

5.1. Desempenho do plano de controle

Nas medições realizadas para o processo de recuperação de falha, ilustrados na Figura 4(a), o primeiro resultado é referente ao sistema sem carga, isto é, quando não há tráfego no plano de controle. Nesse estado, a latência do processo de recuperação de falha foi de 1,87 ms; no início do tráfego de pacotes, com taxa de 250 *packet-in/s*, o tempo atingiu 7,59 ms; com envio de 500 *packet-in/s*, chegou a 18,25 ms; a 750 *packet-in/s*, subiu para 37,68; e por fim, com taxa de 1.000 *packet-in/s*, alcançou 227,22 ms.

No processo de restauração de falha, apresentado na Figura 4(a), a mesma sequência do experimento anterior foi seguida. A latência da restauração de falha, sem tráfego foi de 8.42 ms; com tráfego de 250 *packet-in/s*, subiu para 14,19 ms; com taxas de 500 e 750 *packet-in/s*, aumentou para 21,65 e 45,84 ms respectivamente; já com a taxa de 1.000 *packet-in/s*, atingiu quase 5 s (em média), e com alta variabilidade.

Os resultados para a carga de processamento das RouterBoards, apresentados na Figura 4(b), de acordo com as taxas de geração de pacotes, variaram entre 77,33% e 99%. Com o envio a 250 *packet-in/s*, obteve-se o menor percentual de uso de processador, 77,33%; ao subir a taxa para 500 *packet-in/s*, o uso foi a 84,53%; no terceiro experimento, a taxa de 750 *packet-in/s*, chegou a 96,4%; e por fim, quando a taxa de envio era 1.000

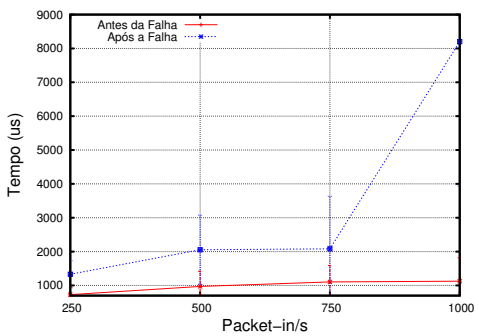
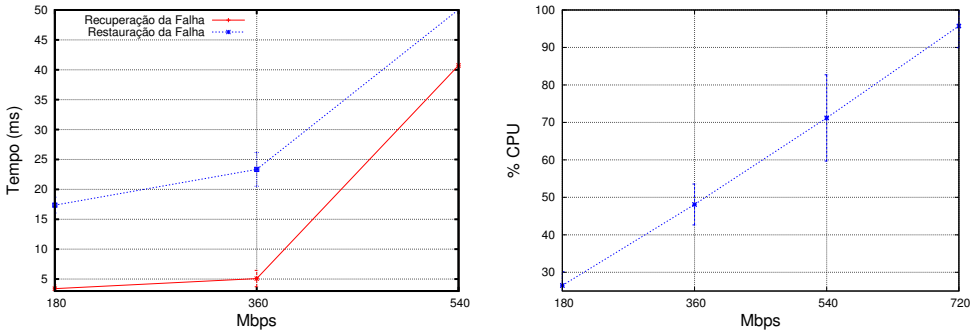


Figura 5. Tráfego no Plano de Controle: Latência de Packet-in.



(a) Recuperação e Restauração da Falha (b) Carga de Processamento nas RouterBoards

Figura 6. Tráfego no Plano de Dados

packet-in/s, alcançou 99% de uso, número próximo a ocupação total do processador.

Em nossos experimentos também avaliamos a latência do processamento de pacotes pelos controladores. Os valores para a latência do processamentos de *packet-in* são apresentados na Figura 5.1. Essa medida varia de acordo com os eventos de falha no plano de controle. Por exemplo, em um dos cenários apresentados, temos uma taxa de 250 *packet-in/s*. Ao ocorrer uma falha, o controlador que continua ativo, passa a receber todo o tráfego que tinha como destino o *faulty-controller*, ou seja, 500 *packet-in/s*. Com início dos testes, a latência de processamento variou entre 729 μs e 1.126 μs, e após a falha de um controlador, entre 1.332 μs a 8.200 μs.

5.2. Desempenho do plano de dados

Após os resultados dos experimentos com tráfego no plano de controle apresentamos os dados dos experimentos com foco na variação de carga no plano de dados.

A sequência dos resultados segue a mesma ordem dos experimentos apresentados anteriormente, e são mostrados na Figura 6(a). O primeiro resultado refere-se a recuperação de falha. Repetindo a abordagem em que o tráfego da rede aumenta de forma gradual, a latência para a recuperação de falha, neste caso, variou entre 3,39 ms e 398,2 ms. No primeiro cenário, com tráfego de 180 Mbps, o tempo para recuperação de falha, foi 3,39 ms; com o aumento da taxa para 360 Mbps, a latência subiu para 5,08 ms; a 540 Mbps, alcançou 40,73 ms; e por fim, com a taxa de 720 Mbps, a máxima foi 398,2 ms.

No processo de restauração de falha, os resultados apresentados estão ilustrados na Figura 6(a): com taxa de 180 Mbps, a latência ficou em 17,37 ms; a 360 Mbps, subiu para 23,34ms; com o aumento do tráfego para 540 Mbps, alcançou 50,06 ms; e ao final, com taxa de 720 Mbps, chegou a 119,28ms.

Na última série de experimentos realizados, exibimos os dados sobre a utilização de CPU das RouterBoards, Figura 6(b). Para os cenários apresentados, o percentual de utilização variou entre 26,48% e 95.73% e seguiu um comportamento linear.

5.3. Comentários finais

Algumas considerações podem ser feitas em relação aos resultados dos experimentos. No geral, o processo de restauração de falha tem maior latência que o de recuperação, e isso deve-se ao fato de que a quantidade de mensagens trocadas no processo de migração de

switches é maior. E mais, algumas dessas mensagens, como a *barrier-message*, indicam que, uma vez recebida, o *switch* deve processar todas as mensagens pendentes antes de qualquer outra mensagem que tenha chegado após a *barrier-request*, o que pode aumentar consideravelmente a latência.

Os *switches*, nos cenários com 1.000 *packet-in/s* e 720 Mpbs, operaram no limite de sua capacidade, o que causou grande variação nos resultados. Isso pode ser observado na variabilidade dos resultados encontrados. Ainda sobre o desempenho das RouterBoards, os resultados apontam que a CPU dos *switches* são mais sensíveis ao encaminhamento de *packet-in* para ao plano de controle do que ao encaminhamento de pacotes no plano de dados. Desde o primeiro cenário, com tráfego no plano de controle, os *switches* apresentaram alta carga de CPU, com 250 *packet-in/s* a taxa de utilização foi de 77,33%. Já nos experimentos com tráfego no plano de dados, a medida que a taxas variaram, o aumento do consumo de CPU se deu de maneira gradual.

Um outro importante levantamento diz respeito a latência de processamento de *packet-in*. Durante os experimentos, as latências aumentaram de forma gradual à medida em que a taxa de pacotes gerados variava, partindo de 729 e chegando a 1.126 μs. Após a falha de um controlador, para os cenários com taxa de 250, 500, 750 *packet-in/s*, o tempo de processamento para cada pacote, praticamente, dobrou; e para 1.000 *packet-in/s* a latência foi basicamente 8 vezes maior, atingindo 8200 μs. Essa análise reforça a importância do plano de controle ser escalável.

6. Trabalhos Relacionados

Há uma miríade de questões de pesquisa em recentes trabalhos relacionados à distribuição de controle em redes SDN objetivando soluções de alta disponibilidade. Nesta seção citamos apenas os trabalhos mais próximos ao foco da proposta deste artigo, que é a solução do problema de conectividade com múltiplos controladores e o uso de *data stores* para consistência do estado.

ONIX [Koponen et al. 2010] foi a proposta pioneira no controle distribuído em redes SDN particionando o escopo dos controladores, agregando as informações, e compartilhando o estado via APIs para diferentes tipos de *data stores* em função dos requisitos de consistência. O trabalho, porém, não discute a conectividade com múltiplos controladores nem o tratamento de falhas, que ficam sob a responsabilidade das aplicações.

Em [Fonseca et al. 2013] os autores apresentam um estudo em SDN em que as estratégias ativa e passiva são usadas para implementar controladores distribuídos. Além disso, indicam cenários em que utilização de cada técnica pode ser mais efetiva. No experimento que adota replicação passiva, os *switches* se conectam a dois controladores: um primário e outro secundário. O controlador primário é responsável por processar todos os pacotes da rede e manter uma cópia de seu estado no secundário. Na implementação utilizando replicação ativa, os *switches* se conectam a todos os controladores da rede, e estes, processam os pacotes de forma simultânea. Neste método, após receber e processar os pacotes, apenas um controlador responde cada requisição, e para tal, trocam mensagem de controle para definir o controlador que processou o pacote no menor tempo. Essa técnica permite que os estados da rede estejam replicados em cada controlador, já que todos controladores processam o mesmo pacote.

Apesar do estudo levantar pontos relevantes para a área, as duas implementações

apresentadas utilizam a versão 1.0 do protocolo OpenFlow, deixando a cargo dos projetistas de SDN implementar seus próprios mecanismos para que *switches* lidem com múltiplos controladores. Outra limitação da estratégia ativa implementada, segundo os autores, é que a sincronização dos controladores usando consenso distribuído aumenta consideravelmente a latência da rede e reduz a capacidade de processamento, já que adicionar controladores a arquitetura não aumenta o *throughput* do plano de controle.

Em [Botelho et al. 2014] apresenta-se uma proposta de uma arquitetura para múltiplos controladores chamada SmartLight. A estratégia é baseada em uma variação da abordagem passiva, em que há um controlador principal, que processa *packet-in*, e *N* controladores *backup*. Também, foi implementado um serviço de coordenação entre os controladores, de modo que em caso de falha no controlador principal, uma das réplicas possa assumir o controle da rede. Um dos pontos centrais do trabalho, é a utilização de um *data store* como meio de distribuir os estados da rede. Apenas o controlador principal interage com o *data store*, escrevendo e recuperando informações, e em caso de falha, o novo controlador principal deve recuperar os estados armazenados.

O trabalho apresenta uma linha promissora, no entanto uso de apenas um controlador principal limita a vazão do plano de controle. Um outro ponto levantado é que, em caso de falha, o controlador que assumir as operações da rede, precisa recuperar todos os estados no *data store*, o que adiciona uma maior latência na transição dos controladores.

7. Conclusão

A implementação de um sistema distribuído tolerante a falhas, tal como o plano de controle de redes SDN, envolve vários *trade-offs*. Neste trabalho, discutimos aqueles relacionados à implementação de controladores replicados ativa e passivamente.

Provavelmente o *trade-off* mais importante neste cenário é o conflito entre consistência forte nos estados das réplicas, o que permite aos diversos controladores atuar prontamente nos *switches*, e o custo em termos de comunicação para implementá-la. Neste trabalho, implementamos e avaliamos uma proposta de plano de controle distribuído usando replicação ativa, que provê consistência forte. Nossa avaliação, em ambiente real, utilizando elementos de rede de prateleira (COTS), o controlador *open-source* Ryu e o serviço OpenReplica, sugere que é viável o uso de consistência forte. Trabalhos relacionados da literatura ficaram limitados a ambientes de emulação/simulação e não focaram nas configurações de papéis no OpenFlow usando replicação via *data stores*.

Em relação aos resultados dos experimentos realizados, observamos o comportamento do ambiente, introduzindo falhas nos controladores da rede com variações de tráfego no plano de controle e no plano de dados. Nas operações de recuperação e restauração de falha, ficou caracterizado que a segunda, em geral, tem maior latência. Já sobre o desempenho dos *switches*, os resultados mostram que o encaminhamento de pacotes para o plano de controle, do ponto de vista da CPU, é mais custoso que o encaminhamento no plano de dados.

Para trabalhos futuros, pretendemos utilizar a capacidade de troca de controladores de *switches* para explorar o balanceamento de carga entre controladores. Na mesma linha, pretendemos habilitar os controladores a trabalhar com uma visão parcial da rede, compatível com a visão global, de forma a reduzir a carga em cada controlador.

Referências

Botelho, F. A., Bessani, A. N., Ramos, F. M. V., and Ferreira, P. (2014). Smartlight: A practical fault-tolerant SDN controller. *CoRR*, abs/1407.6062.

Botelho, F. A., Ramos, F. M. V., Kreutz, D., and Bessani, A. N. (2013). On the feasibility of a consistent and fault-tolerant data store for sdns. In *Software Defined Networks (EWSDN), 2013 Second European Workshop on*, pages 38–43. IEEE.

Budhiraja, N., Marzullo, K., Schneider, F. B., and Toueg, S. (1993). The primary-backup approach. *Distributed systems*, 2:199–216.

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

Dixit, A., Hao, F., Mukherjee, S., Lakshman, T., and Kompella, R. (2013). Towards an elastic distributed sdn controller. *SIGCOMM Comput. Commun. Rev.*, 43(4):7–12.

Fonseca, P., Bennesby, R., Mota, E., and Passito, A. (2013). Resilience of sdns based on active and passive replication mechanisms. In *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 2188–2193.

Heller, B., Sherwood, R., and McKeown, N. (2012). The controller placement problem. In *HotSDN '12*, pages 7–12, New York, NY, USA. ACM.

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

Koponen et al. (2010). Onix: A distributed control platform for large-scale production networks. *OSDI, Oct*.

Kreutz, D., Ramos, F. M. V., Veríssimo, P., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):63.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565.

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

Liberato, A. B., Mafioletti, D. R., Spalla, E. S., Martinello, M., and Villaca, R. S. (2014). Avaliação de desempenho de plataformas para validação de redes definidas por software. *Wperfomance - XIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, pages 1905–1918.

Panda, A., Scott, C., Ghodsi, A., Koponen, T., and Shenker, S. (2013). Cap for networks. In *HotSDN '13*, pages 91–96, New York, NY, USA. ACM.

Penna, M. C., Jamhour, E., and Miguel, M. L. (2014). A Clustered SDN Architecture for Large Scale WSON. In *AINA*, pages 374–381. IEEE.

Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., and Magalhães, M. F. (2010). OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. *Cad. CPqD Tecnologia, Campinas*, 7(1):65–76.

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

Melhorando a Precisão e Repetibilidade de Experimentos no PlanetLab

Lauro Luis Costa, Luis C. E. Bona, Elias P. Duarte Jr.

[1]Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19018 – 81531-980 – Curitiba – PR – Brazil

{llcosta,bona,elias}@inf.ufpr.br

Abstract. *Current Internet users and applications require high levels of performance, security, scalability and mobility. These characteristics were not predicted at the time of its creation. Large-scale testbeds like PlanetLab have been developed to allow the design and realistic evaluation of protocols and architectures to supply the new Internet demands. However, it is particularly hard to guarantee the precision and repeatability of the results obtained. In this work, we present an experimental evaluation of the impact of using subsets of nodes selected with a node connectivity monitoring tool on the precision and repeatability of the results obtained. Applications with different profiles were executed. Results show that the selection strategy based on* k-cores *reduces the variation on the results obtained by up to 27% and resulted on an average execution time up to 28% faster compared to other alternatives.*

Resumo. *Usuários e aplicações da Internet atualmente demandam características de desempenho, segurança, escalabilidade e mobilidade que não foram previstas no momento de sua criação.* Testbeds *de larga escala, como o PlanetLab, são peças chave para o desenvolvimento e avaliação de novas aplicações e arquiteturas que atendam a estas demandas. No entanto, um dos principais desafios no PlanetLab é garantir a precisão e repetibilidade dos resultados obtidos. Neste trabalho, apresentamos uma avaliação experimental do impacto da utilização de subconjuntos de nodos selecionados por uma ferramenta de monitoramento de conectividade na precisão e repetibilidade dos resultados obtidos. Foram realizados experimentos utilizando aplicações com diferentes perfis de utilização executadas por subconjuntos de nodos obtidos por estratégias de seleção distintas. A estratégia de seleção de nodos* k-core *reduziu a variação dos resultados obtidos em até 27% e obteve médias de tempo de execução até 28% mais baixa que uma das estratégias alternativas.*

1. Introdução

Desde seu surgimento a Internet vem crescendo e evoluindo rapidamente, de uma rede acadêmica para uma ampla plataforma de informação, interação social e comércio, afetando organizações e indivíduos nas mais diversas formas. As aplicações desenvolvidas, e seu uso, mudaram drasticamente a natureza do tráfego na rede. As novas aplicações demandam características de desempenho, segurança, escalabilidade e mobilidade que não foram previstas na criação da Internet [Pan et al. 2011]. Os *testbeds* de larga escala são

essenciais para a validação de novos protocolos e serviços, entre os principais exemplos de *testbeds* estão as redes GENI[1], FIRE[2], FIBRE[3] e PlanetLab[4] [Chun et al. 2003].

A rede do *testbed* PlanetLab é formada por uma coleção de máquinas mantidas por instituições de pesquisa. Estas máquinas estão amplamente distribuídas ao redor do mundo, sendo atualmente cerca de 1333 nodos em 634 locais diferentes. O usuário acessa um *slice*, que é uma abstração para recursos disponíveis nos nodos da rede. Este é um ambiente altamente instável [Duarte Jr. et al. 2010], apresentando grandes variações no tempo de resposta e largura de banda disponível para a comunicação entre os nodos, e também variações na disponibilidade de recursos em cada nodo. Apesar destas variações refletirem o comportamento da própria Internet, muitas vezes pesquisadores precisam de maior controle sobre estas variações.

Não é trivial encontrar um grupo de nodos estáveis entre si no PlanetLab. Neste trabalho, utilizamos a ferramenta de seleção de nodos estáveis baseado no monitoramento da comunicação par-a-par proposto em [Bona et al. 2012]. Diferentemente do CoMon [Park and Pai 2006] e SWORD [Albrecht et al. 2008], que monitoram informações de cada nodo individualmente, esta ferramenta de seleção monitora a estabilidade da conexão par-a-par entre todos os nodos da rede.

Este trabalho apresenta uma avaliação experimental do impacto da seleção de nodos na precisão dos experimentos. Também pretende-se demonstrar que o compartilhamento dos parâmetros de seleção de nodos utilizados na ferramenta de seleção é suficiente para a obtenção de um conjunto de nodos com características de estabilidade similar, contribuindo para a repetibilidade e reprodutibilidade dos experimentos. Nos experimentos realizados neste trabalho, os nodos selecionados pela ferramenta utilizada apresentaram melhor desempenho e maior consistência nos resultados obtidos. A média do tempo de execução das aplicações foi até 26% mais baixa que o resultado utilizado por um conjunto de nodos selecionados com uma estratégia alternativa. A variação entre os resultados obtidos durante um período de 40 dias foi avaliada, e os nodos selecionados pela ferramenta de monitoramento obtiveram um coeficiente de variação mais baixo quando comparados aos resultados utilizando nodos da estratégia alternativa.

Os resultados obtidos nos experimentos realizados também demonstraram que a ferramenta de monitoração da estabilidade das conexões fim-a-fim é adequada tanto para aplicações com intensa troca de mensagens quanto aplicações distribuídas com intenso uso de CPU. Os resultados demonstram ainda que a ferramenta é capaz de selecionar conjuntos de nodos com características de estabilidade equivalente, apenas utilizando os mesmos parâmetros de seleção.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta trabalhos relacionados, a Seção 3 descreve nossa proposta de utilização da ferramenta de seleção de nodos estáveis para melhorar a precisão e repetibilidade de experimentos no PlanetLab. A Seção 4 apresenta uma avaliação experimental da utilização dos subconjuntos de nodos selecionados, seguida da conclusão na Seção 5.

[1]http://www.geni.net
[2]http://www.ict-fire.eu
[3]http://fibre-ict.eu
[4]https://www.planet-lab.org

2. Trabalhos Relacionados

O planejamento de experimentos é uma etapa fundamental no processo de avaliação de desempenho de um sistema através da experimentação [Barros Neto and Scarminio 2002]. Mesmo quando um experimento é repetido sob condições tão semelhantes quanto possível, é comum a ocorrência de flutuações nos resultados de cada repetição. Essa variação é chamada *erro experimental* ou *ruído* [Box et al. 2005].

Os autores do PlanetLab [Spring et al. 2006] afirmam que os resultados dos experimentos no PlanetLab não são reproduzíveis, e que o *testbed* foi projetado para oferecer condições reais de rede e não um ambiente controlável. Entretanto, no caso de aplicações que executam por longos períodos de tempo, os autores sugerem que pesquisadores devem ser capazes de detectar padrões no comportamento da rede e entender o desempenho e confiabilidade de seus experimentos. No caso de aplicações de curta duração, os autores sugerem a seleção de nodos utilizando a ferramenta CoMon.

O CoMon [Park and Pai 2006] é um serviço de monitoramento desenvolvido para o PlanetLab que oferece aos seus usuários informações referentes aos nodos da rede. Cada nodo da rede monitora diversas informações, incluindo *uptime*, se o nodo está respondendo a tentativas de conexão SSH (*Secure Shell*), uso de processador, uso de memória, uso de disco, entre outros. As informações obtidas pelos nodos são mantidas em um servidor central. O usuário pode fazer consultas neste servidor, utilizando diversos parâmetros, e obter uma lista de nodos que apresentem características desejadas para a realização de um experimento, ou ainda nodos problemáticos a serem evitados. O projeto CoMon esteve operacional e disponível para consultas desde 2004, mas foi encerrado em 2012 devido à dificuldades de compatibilidade e manutenção do sistema.

O SWORD [Albrecht et al. 2008] é uma ferramenta de seleção de nodos no PlanetLab que permite localizar um conjunto de nodos que atendam a uma lista de parâmetros desejados. O SWORD se comunica com outras ferramentas de monitoramento do PlanetLab, incluindo o CoMon, para obter informações sobre os nodos. Além das características de cada nodo, como carga do processador, uso de memória, disco e outros, o usuário pode requisitar nodos que atendam parâmetros de comunicação entre si, como largura de banda e tempo de resposta. A interface da ferramenta oferece ao usuário uma lista com diversos parâmetros relacionados aos recursos disponíveis nos nodos e à conexão entre os nodos. O usuário escolhe quais parâmetros são relevantes em sua consulta e define valores mínimos e máximos para estes parâmetros. Devido à indisponibilidade do CoMon, atualmente não é possível selecionar nodos utilizando a ferramenta SWORD.

MyOps[5] é uma ferramenta de monitoramento do próprio PlanetLab. Esta ferramenta é responsável por verificar se os nodos da rede estão funcionando corretamente e enviar notificações automáticas aos responsáveis caso algum problema seja detectado. Se um nodo fica *offline* por um determinado período de tempo ou é detectada a presença de um *firewall* bloqueando as principais portas, uma notificação é enviada à instituição responsável pelo nodo para que as devidas medidas sejam aplicadas e o nodo volte a funcionar de maneira adequada. O MyOps disponibiliza um portal com uma lista pública do estado dos nodos monitorados e permite ao usuário executar buscas por nodos e obter

[5]http://monitor.planet-lab.org/monitor/node e http://monitor.planet-lab.eu/monitor/node

informações relacionadas, como estado (*online* ou *offline*), estado do SSH e *uptime*.

A execução de experimentos repetíveis em condições realísticas para redes sem fio é abordada em [Lenders and Martonosi 2009]. Experimentos com comunicação sem fio são difíceis de repetir com precisão e modelos de simulação abstratos utilizam parâmetros teóricos não realísticos. Os autores apresentam modelos para a camada física e MAC de redes sem fio baseados em medições realizadas em redes reais, para serem utilizados em simulações e emulações.

Um *framework* para a automatização do projeto e execução de experimentos chamado Weevil é apresentado em [Y. et al. 2008]. O trabalho foca-se na automatização da geração de cargas de trabalho e controle de parâmetros para a obtenção de resultados representativos e repetíveis. Os autores ressaltam a necessidade de controle sobre o ambiente de experimentação e as dificuldades de se obter tal controle no PlanetLab.

3. Seleção de Nodos Estáveis para a Precisão e Repetibilidade de Experimentos no PlanetLab

Esta seção descreve de forma sucinta a ferramenta de seleção de nodos utilizada neste trabalho [Bona et al. 2012]. A arquitetura da ferramenta de seleção de nodos baseada na monitoração de estabilidade das interações fim-a-fim é composta por três módulos: módulo de monitoramento, módulo servidor e módulo cliente. Estes módulos são responsáveis, respectivamente, pelo monitoramento das interações entre os nodos, armazenamento centralizado das informações de monitoramento e interface de interação com o usuário.

Cada nodo do PlanetLab executa um *daemon* de monitoramento. Estes *daemons* se comunicam entre si, enviando e respondendo mensagens utilizando um *socket* UDP. Cada *daemon* deve enviar, periodicamente, mensagens para outros nodos. O tempo até a resposta de cada mensagem (RTT) é observado e armazenado. Um valor de *timeout* é utilizado e caso uma resposta não seja obtida antes do *timeout*, o nodo é considerado indisponível. É importante notar que o RTT é medido na camada de aplicação, portanto o tempo de resposta é influenciado não apenas pelas características de rede, mas também pela carga de processamento no nodo. O módulo servidor é responsável por armazenar os dados obtidos pelo módulo de monitoramento e oferecer dados ao módulo cliente. O servidor armazena em seu banco de dados as informações obtidas na última hora de monitoramento e versões sumarizadas destes dados.

O módulo cliente oferece ao usuário uma interface onde ele pode selecionar um intervalo de tempo e um limiar de RTT, para obter como resposta, uma lista de pares de nodos que apresentaram um tempo de resposta entre si abaixo do limiar selecionado durante o período de tempo desejado. A interface exibe a lista de pares de nodos obtidos e também permite ao usuário selecionar um subconjunto destes nodos utilizando diferentes estratégias de seleção.

Uma lista de pares de nodos que apresentam tempo de resposta entre si abaixo de um certo limiar em um período de tempo é na verdade uma lista de arestas que forma um grafo, onde cada nodo é representado por um vértice, e uma aresta ligando dois vértices indica que o RTT entre os nodos representados está abaixo de um certo limiar no intervalo de tempo selecionado. A presença de um vértice representando um nodo neste grafo não significa que o nodo faz parte de um subconjunto de nodos estáveis. É desejável encontrar

um subconjunto de nodos que apresente um alto grau de conectividade entre si, ou seja, um subgrafo onde os vértices possuem um alto grau de conectividade. Para isto, diferentes estratégias de seleção podem ser utilizadas, como a estratégia do Grau Mínimo, Maior Grau Mínimo, *k-Core* e Clique Estável. Os experimentos realizados pelos autores em [Bona et al. 2012] demonstraram que a melhor estratégia a ser utilizada é o *k-Core*.

4. Resultados Experimentais

Esta seção descreve os resultados experimentais obtidos na avaliação da repetibilidade de experimentos no *testbed* PlanetLab. A métrica utilizada para avaliar precisão dos experimentos realizados é o *coeficiente de variação* (desvio padrão dividido pela média). O coeficiente de variação é proporcional à média, portanto é adequado para a comparação da variação entre o resultados de amostras com médias diferentes.

Foram implementadas três aplicações distribuídas para serem executadas no ambiente do PlanetLab. As aplicações escolhidas apresentam diferentes características de uso de recursos, variando de intensa troca de mensagens e reduzido uso de CPU a intenso uso de CPU e reduzida troca de mensagens. Foram obtidos subconjuntos de nodos utilizando a estratégia de seleção descrita na Seção 3 e também através de uma estratégia alternativa. Para cada aplicação, a precisão dos resultados obtidos utilizando cada subconjunto de nodos é comparada utilizando o coeficiente de variação.

Os experimentos foram organizados em *rodadas de experimentos* e *séries*. Uma série é composta por 30 rodadas de experimentos e cada rodada de experimentos consiste na execução das três aplicações utilizando cada um dos subconjuntos de nodos. Uma rodada de experimentos se completa em aproximadamente 150 minutos, a próxima rodada inicia-se automaticamente logo ao fim da anterior, sem nenhuma restrição de horário ou dia da semana. As séries são iniciadas manualmente e têm duração de aproximadamente 3 dias. No total, foram executadas 300 rodadas de experimentos organizadas em 10 séries, sendo a primeira série iniciada em 20 de abril de 2014 e a última em 25 de maio de 2014.

Esta seção está organizada da seguinte maneira. A Subseção 4.1 descreve as estratégias e os parâmetros utilizados na seleção dos subconjuntos de nodos utilizados para a realização dos experimentos. A Subseção 4.2 apresenta detalhes das aplicações utilizadas nos experimentos e os resultados obtidos. A Subseção 4.3 apresenta uma análise geral dos resultados dos experimentos realizados.

4.1. Seleção dos Subconjuntos de Nodos

Apesar de o PlanetLab ser formado por mais de mil nodos, o número de nodos acessíveis via SSH durante a execução destes experimentos foi muito mais baixo, usualmente em torno de 400 nodos. Para a execução dos experimentos, foram utilizados quatro subconjuntos de 150 nodos. Dois subconjuntos foram selecionados utilizando a estratégia de seleção de nodos apresentada na Seção 3, chamados `core-150ms` e `core-150ms-everyround`. Os outros dois subconjuntos, obtidos utilizando uma estratégia alternativa, são chamados `ping.c3sl` e `ping.uk`.

Os subconjuntos *k-core* (`core-150ms` e `core-150ms-everyround`) foram selecionados utilizando os seguintes parâmetros: 150 nodos, dados de monitoramento obtidos na última hora, limiar do tempo de resposta de 150ms e estratégia de seleção *k-core*. Uma seleção de nodos utilizando estes parâmetros usualmente resulta em um

subconjunto de nodos com grau mínimo próximo a 120, significando que todos os nodos do subconjunto têm conexão com RTT menor que 150ms para pelo menos outros 120 nodos dentro do subconjunto selecionado.

O subconjunto `core-150ms` foi obtido antes do início dos experimentos, no dia 20 de abril de 2014. Este subconjunto permanece inalterado durante a execução de todos os experimentos. Já o subconjunto `core-150ms-everyround` é obtido novamente no início de cada *rodada de experimentos*. Os nodos são obtidos sempre utilizando os mesmos parâmetros de seleção.

Durante uma das séries de experimentos realizados, a composição do subconjunto `core-150ms-everyround` foi observada. Em um período de aproximadamente três dias, tempo necessário para completar a série, um total de 262 nodos distintos fizeram parte do subconjunto `core-150ms-everyround` em ao menos uma das 30 rodadas de experimento. Destes 262 nodos, 59 foram constantemente selecionados em todas as rodadas e 123 foram selecionados mais de 20 vezes. Por outro lado, 42 nodos foram selecionados 5 vezes ou menos e apenas 11 foram selecionados uma única vez.

Para comparação, selecionamos outros dois subconjuntos de nodos com uma estratégia alternativa, utilizando os valores de tempos de resposta obtidos pela troca de mensagens *Echo Request* e *Echo Reply* do protocolo ICMP, implementada pelo comando *ping*. Nesta estratégia, um nodo central é escolhido e é construída uma lista com os tempos de resposta obtido pela média de cinco execuções do comando *ping* de cada nodo para o nodo central escolhido. Os nodos que obtiverem os valores de tempo de resposta mais baixos são selecionados. Ambos os subconjuntos foram selecionados antes do início dos experimentos, em 20 de abril de 2014, e permanecem inalterados durante as execuções de todos os experimentos. A ferramenta CoMon monitorava o tempo de resposta entre os nodos e seu servidor, esta informação permitia ao usuário selecionar um subconjunto de nodos com os menores tempos de resposta para um servidor central, de forma muito similar a estratégia alternativa descrita.

O primeiro subconjunto de nodos obtidos através desta estratégia, chamado `ping.c3sl`, utilizou como nodo central o nodo *planetlab2.c3sl.ufpr.br*, um dos dois nodos mantidos no PlanetLab pelo Centro de Computação Científica e Software Livre na Universidade Federal do Paraná. O segundo subconjunto, chamado `ping.uk`, utilizou como nodo central o *planetlab2.xeno.cl.cam.ac.uk*. Este nodo foi escolhido por estar constantemente entre os nodos estáveis selecionados utilizando a estratégia de seleção *k-core*, portanto este subconjunto deve conter muitos dos nodos também presentes no subconjunto `core-150ms`.

O subconjunto `ping.uk` é bastante semelhante ao subconjunto `core-150ms`, dos 150 nodos selecionados, 100 estão presentes nos dois subconjuntos. Já o subconjunto `ping.c3sl` é bastante diferente dos outros subconjuntos, apenas 11 de seus nodos também estão presentes no subconjunto `ping.uk` e 27 estão presentes no subconjunto `core-150ms`. Portanto, o subconjunto `ping.uk` é bastante similar aos subconjuntos obtidos a partir da estratégia *k-core*, enquanto o subconjunto `ping.c3sl` é bastante diferentes dos outros três subconjuntos.

4.2. Aplicações Executadas e Resultados

Os experimentos foram realizados no período entre 20 de abril e 28 de maio de 2014. Os experimentos foram executados de forma serializada, alternando os subconjuntos de nodos e as aplicações utilizadas. É importante destacar que, apesar de nossos experimentos serem realizados individualmente, a todo momento existem *slices* pertencentes a outros usuários concorrendo pelos recursos dos nodos.

Para a realização dos experimentos foram utilizadas três aplicações. Duas destas aplicações foram implementadas utilizando o *framework* Map-Reduce [Dean and Ghemawat 2004] e a terceira aplicação utiliza o protocolo *BitTorrent*[6]. As duas aplicações Map-Reduce foram desenvolvidas utilizando o Mincemeatpy[7], uma implementação *leve* do Map-Reduce em Python. O Mincemeatpy permite o desenvolvimento de aplicações simples e não requer a instalação e configuração complexa antes da execução de um experimento.

Uma aplicação Mincemeatpy possui um único nodo *master* e vários *workers*. O nodo *master* organiza as funções *Map* e *Reduce* como tarefas e os *workers* se conectam ao *master* e executam tarefas assinaladas. Nesta implementação, os *workers* não se comunicam entre si, toda comunicação é gerenciada pelo nodo *master*.

As subseções seguintes apresentam mais detalhes sobre as características de cada aplicação e os resultados obtidos nos experimentos realizados com cada uma delas. A primeira subseção apresenta a aplicação *Triangular.py*, a subseção seguinte apresenta a aplicação *Bit.py* e a terceira subseção apresenta a aplicação *Torrent*.

4.2.1. Aplicação Map-Reduce 1: *Triangular.py*

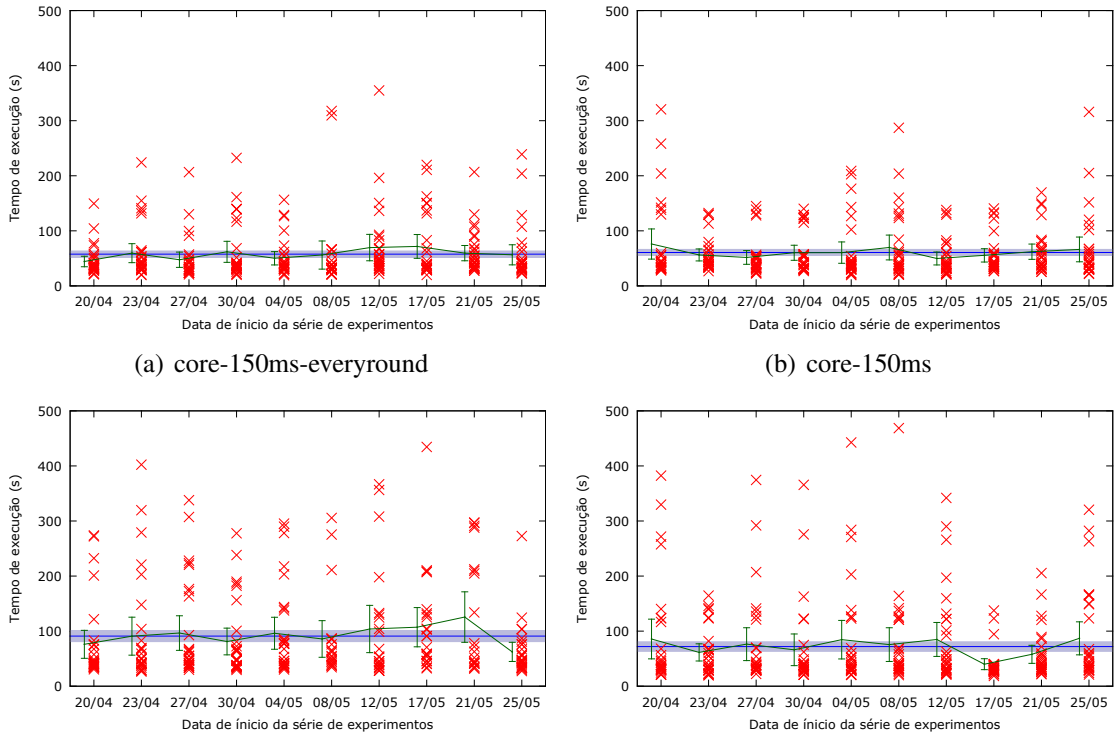
A primeira aplicação desenvolvida utilizando o Mincemeatpy, *Triangular.py*, retorna uma lista com os N primeiros números triangulares. O n-ésimo número triangular T_n é dado por $(n^2+n)/2$, para n pertencente ao conjunto dos números naturais. Com os parâmetros selecionados para esta sequência de experimentos, a aplicação calcula os primeiros 8192 números triangulares. A implementação desta aplicação não é a forma mais eficiente de se encontrar números triangulares, mas tem como característica o baixo uso de CPU e intensa troca de mensagens. Para calcular os primeiros 8192 números triangulares, a função *map* emite mais de 33 milhões de pares (chave,valor)

Os gráficos na Figura 1 apresentam os resultados obtidos com a execução da aplicação *Triangular.py*. Foram realizadas 300 rodadas de experimentos, organizadas em 10 séries de 30 repetições da aplicação para cada subconjunto. O eixo y indica o tempo de execução em segundos e o eixo x indica a data de execução de cada série de experimentos. A linha azul cruzando o gráfico representa a média do tempo de execução destas 300 repetições, e a sombra cinza o intervalo de confiança no nível de 95%. A linha verde apresenta a média e intervalo de confiança no nível de 95% para cada série de 30 repetições. Cada *x* vermelho representa o tempo de execução de um experimento.

Observando a distribuição dos *x* vermelhos, nota-se a presença de diversos valores muito distantes da média nos quatro gráficos, como por exemplo os dois resultados acima

[6]http://www.bittorrent.org/beps/bep_0003.html

[7]https://github.com/michaelfairley/mincemeatpy



(a) core-150ms-everyround (b) core-150ms

(c) ping.c3sl (d) ping.uk

Figura 1. Tempos de execução da aplicação *Triangular.py*.

de 300 segundos obtidos pelo subconjunto `core-150ms-everyround` na série iniciada no dia 08/05. Porém, os valores distantes da média aparecem com maior frequência para os subconjuntos obtidos com a estratégia *ping*. Observando-se a linha verde, comparando as médias obtidas em cada série de 30 repetições e a média geral obtida por todas as 300 repetições do experimento, a distância entre estas médias é menor nos resultados obtidos com os subconjuntos *k-core* comparados aos resultados obtidos com os subconjuntos *ping*.

Subconjunto	core-150ms everyround	core-150ms	ping.c3sl	ping.uk
Média (s)	52,79	56,36	85,08	65,59
Desvio Padrão (s)	37,26	36,65	71,57	60,09
Mínimo (s)	21,02	21,69	29,13	20,95
Máximo (s)	206,89	202,69	307,26	320,42
Coeficiente de variação	0,705	0,650	0,841	0,916

Tabela 1. Sumarização dos resultados para a aplicação *Triangular.py*.

A Tabela 1 apresenta informações referentes aos tempos de execução de todas as 300 repetições da aplicação *Triangular.py* com cada subconjunto de nodos, excluindo-se 5% dos dados mais discrepantes, os 2,5% resultados mais altos e mais baixos obtidos. Todos os subconjuntos apresentaram um coeficiente de variação bastante alto, mas a estratégia de seleção *k-core* contribuiu para o aumento da precisão, obtendo coeficientes de variação mais baixos, comparados aos subconjuntos *ping*. Observando a última linha da tabela, o coeficiente de variação dos subconjuntos *k-core* é de 0,705 e 0,650, enquanto os

subconjuntos *ping* obtiveram 0,841 e 0,916.

Já observando-se as médias de tempo de execução, é possível perceber que os subconjuntos *k-core* obtiveram um tempo de execução médio mais baixo que os subconjuntos *ping*. Considerando os resultados eliminando 5% dos dados discrepantes, as médias de tempo de execução dos subconjuntos *k-core* foram de 52,79s e 56,36s, contra as médias de 85,08s e 65,59s obtidas pelos subconjuntos *ping*.

Em geral, os subconjuntos obtidos com a estratégia *k-core* resultaram em tempo de execuções mais baixos e resultados mais consistentes do que os subconjuntos obtidos com a estratégia *ping*. O subconjunto `core-150ms-everyround` obteve uma média de tempo de execução mais baixa que o subconjunto `core-150ms`. A seleção de novos nodos no início de cada rodada de experimento força a escolha de novos nodos para o experimento, o que deveria implicar em aumento do coeficiente de variação, no entanto o subconjunto `core-150ms-everyround` obteve um coeficiente de variação próximo ao do subconjunto `core-150ms`, demonstrando a capacidade da estratégia *k-core* de selecionar subconjuntos de nodos com características de conectividade semelhante ao longo do tempo.

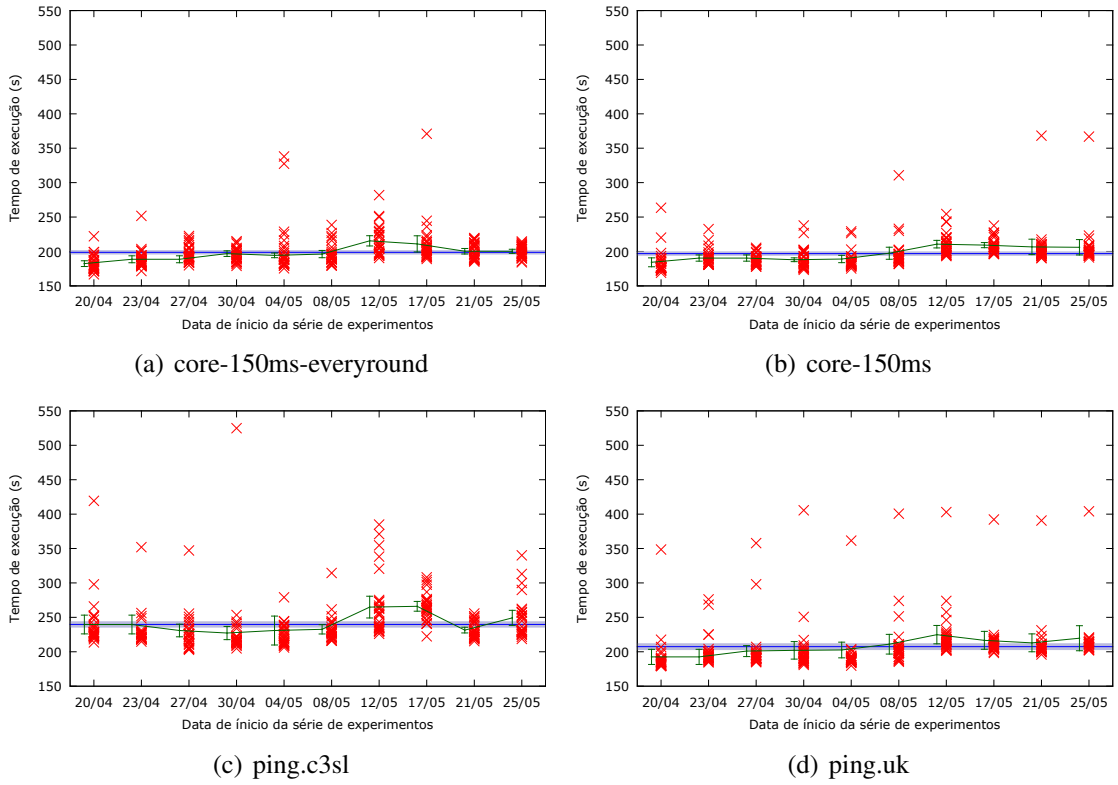
4.2.2. Aplicação Map-Reduce 2: *Bit.py*

A segunda aplicação desenvolvida foi inspirada no problema resolvido por *mineradores* da cripto moeda Bitcoin[8]. O Bitcoin, lançado em 2009, é uma moeda com criação e transferência baseada em protocolos de código aberto de criptografia. A segurança e confiabilidade desta moeda vem da dificuldade de se validar uma transação incorreta. As transações de Bitcoins são organizadas em blocos e validadas por uma rede P2P. A validação destas transações requer grande poder de processamento, mas gera recompensa financeira aos usuários envolvidos, chamados *mineradores*.

A validação de um bloco de transações ocorre utilizando a função *hash* SHA-256 [Eastlake 3rd 2001]. De forma simplificada, o objetivo de um minerador de Bitcoin é, dado um valor de entrada que representa o bloco de transações e informações relacionadas ao bloco anterior, encontrar um valor v tal que SHA-256 $(entrada + v) < h$, sendo h um valor em hexadecimal de 256 bits, usualmente com muitos zeros à esquerda. A aplicação implementada resolve uma versão simplificada deste problema. Com os parâmetros selecionados, durante uma execução da aplicação, a função SHA-256 é calculada para quase 7 bilhões de valores de v. Destes bilhões de valores para v, mais de 6 mil resultaram em 5 zeros à esquerda na *hash* resultante, mais de 300 com 6 zeros, 36 com 7 zeros e apenas 2 com 8 zeros.

Os gráficos na Figura 2 apresentam os resultados obtidos com a execução da aplicação *Bit.py*, seguindo a mesma organização utilizada nos gráficois da aplicação anterior. Observando a distribuição dos *x* vermelhos, nota-se a presença de diversos valores muito distantes da média nos resultados dos subconjuntos utilizando a estratégia *ping*. Os resultados dos subconjuntos obtidos utilizando a estratégia *k-core* apresentam menor dispersão. Por exemplo o gráfico do subconjunto `core-150ms- everyround` apresenta alguns resultados mais distantes da média nos dias 4 e 17 de maio, já os dois subconjuntos *ping*

[8]https://bitcoin.org/en/



(a) core-150ms-everyround (b) core-150ms

(c) ping.c3sl (d) ping.uk

Figura 2. Tempos de execução da aplicação Bit.py.

apresentam algum resultados distante da média em todas as séries de experimentos.

Observando-se a linha verde nos quatro gráficos, nota-se um comportamento comum aos quatro subconjuntos, com resultados melhores nas primeiras séries de experimentos, e uma piora nos experimentos realizados após o dia 12/05. Esta alteração indica alguma variação no desempenho da rede do PlanetLab, afetando todos os nodos. No entanto, o impacto deste comportamento foi mais suave nos resultados dos subconjuntos *k-core*. Ainda sobre os subconjuntos *k-core*, destaca-se um intervalo de confiança menor, tanto para cada série de experimentos (em verde), quanto para as 300 repetições combinando os resultados das 10 séries (sombra cinza).

Subconjunto	core-150ms everyround	core-150ms	ping.c3sl	ping.uk
Média (s)	196,86	195,02	236,39	202,82
Desvio Padrão (s)	13,21	13,09	22,14	18,86
Mínimo (s)	174,89	175,31	206,37	180,95
Máximo (s)	238,54	237,70	338,23	348,51
Coeficiente de variação	0,067	0,067	0,093	0,092

Tabela 2. Sumarização dos resultados para a aplicação *Bit.py*.

A Tabela 2 apresenta os resultados para a aplicação *Bit.py*, seguindo a mesma organização utilizada na tabela anterior. Esta aplicação resultou em coeficientes de variação mais baixos nos resultados dos quatro subconjuntos, mas a estratégia de seleção *k-core* obteve maior precisão com coeficientes de variação relativamente mais baixos. Obser-

vando a última linha da tabela, o coeficiente de variação dos subconjuntos *k-core* é de 0,067 para ambos, enquanto os subconjuntos *ping* obtiveram 0,093 e 0,092.

Já observando-se as médias de tempo de execução, é possível perceber que os subconjuntos *k-core* obtiveram um tempo de execução médio mais baixo que os subconjuntos *ping*. Considerando os resultados eliminando 5% dos dados discrepantes, as médias de tempo de execução dos subconjuntos *k-core* foram de 195,86s e 195,02s, contra as médias de 236,39s e 202,82s obtidas pelos subconjuntos *ping*.

Os subconjuntos `core-150ms` e `core-150ms-everyround` exibiram comportamento bastante similar entre si e superiores aos subconjuntos `ping.c3sl` e `ping.uk` na média de tempo de execução. O coeficiente de variação foi baixo em todos os subconjuntos e ligeiramente inferior para os subconjuntos *k-core*.

4.2.3. Aplicação 3: *Torrent*

A terceira aplicação utilizada para os experimentos apresentados neste trabalho utiliza uma implementação do protocolo *BitTorrent*, o Transmission[9], software padrão nas distribuições Ubuntu atuais. O protocolo *BitTorrent* suporta o compartilhamento de arquivos em redes P2P. A primeira versão do protocolo foi apresentada em 2001 e, atualmente, é uma das aplicações mais comuns para transferência de arquivos.

Para a execução dos experimentos com *torrents* apresentados nesta Seção, foi utilizado um arquivo gerado com dados aleatórios de tamanho 27MB. Este arquivo está, inicialmente, disponível em apenas um *host*, o tempo entre o início do experimento e a obtenção de uma cópia completa do arquivo foi observada em cada nodo. A métrica utilizada para avaliar a execução do experimento foi a média do tempo observado nos 80 primeiros nodos a completarem a transferência do arquivo.

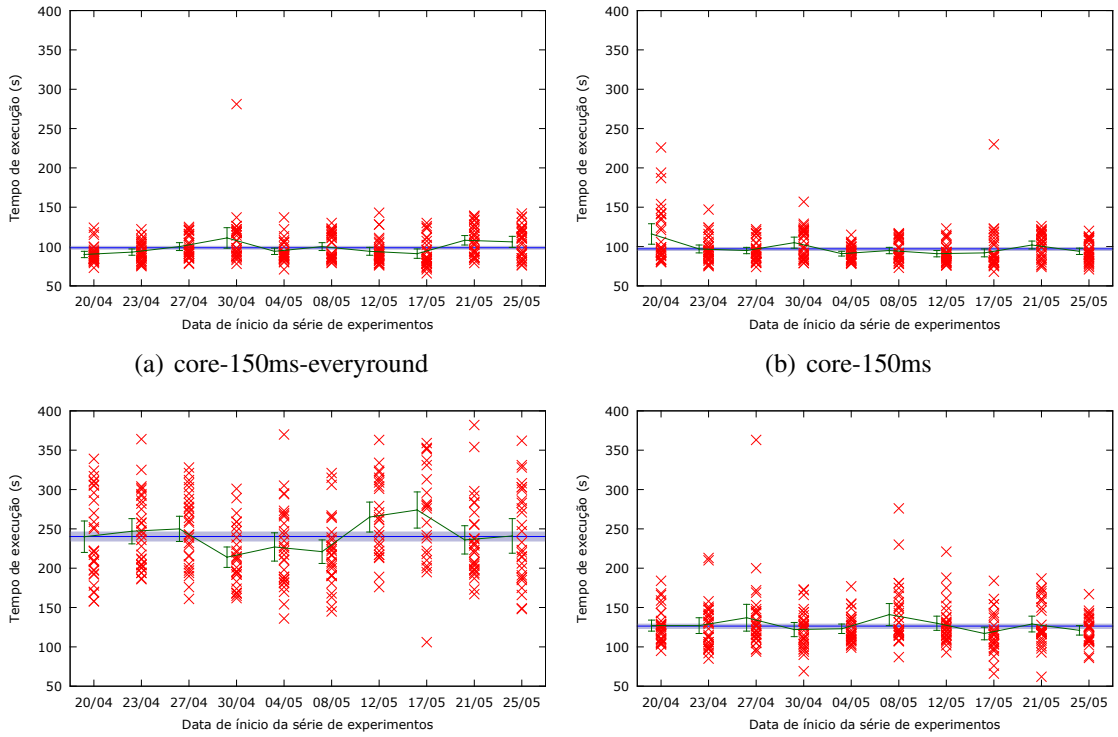
Os gráficos na Figura 3 apresentam os resultados obtidos com a execução da aplicação *Torrent*, seguindo a mesma organização utilizada nos gráficos das aplicações anteriores. Observando-se a distribuição dos *x* vermelhos, é possível notar a grande dispersão das amostras para os subconjuntos da estratégia *ping*. Os subconjuntos *k-core* apresentam amostras mais concentradas e próximas da média geral obtida, com exceção da série realizada no dia 20/04 utilizando o subconjunto `core-150ms`, que apresentou uma dispersão nas amostras incomum às outras séries.

Subconjunto	core-150ms everyround	core-150ms	ping.c3sl	ping.uk
Média (s)	98,46	97,09	240,25	126,37
Desvio Padrão (s)	15,84	15,07	47,67	21,63
Mínimo (s)	75,83	75,93	158,77	88,06
Máximo (s)	137,97	142,30	354,43	188,35
Coeficiente de Variação	0,161	0,155	0,198	0,171

Tabela 3. Sumarização dos resultados para a aplicação *Torrent*.

A Tabela 3 apresenta os resultados para a aplicação *Torrent*, seguindo a mesma organização utilizada nas tabelas anteriores. Observando-se o coeficiente de variação, é

[9]http://www.transmissionbt.com



(a) core-150ms-everyround (b) core-150ms

(c) ping.c3sl (d) ping.uk

Figura 3. Tempos de execução da aplicação Torrent.

possível notar que os subconjuntos *k-core* obtiveram um coeficiente de variação próximos entre si, 0,161 e 0,155, e superiores aos subconjuntos *ping*, 0,198 e 0,171.

A média de tempo obtida pelos subconjuntos *k-core* também são muito semelhantes entre si, 98,46s e 97,09s, superiores à média de 126,37s obtida pelo subconjunto `ping.uk`. Já o subconjunto `ping.c3sl` apresentou um desempenho muito inferior, com uma média de 240,25s, evidenciando uma baixa conectividade entre os nodos do subconjunto.

4.3. Discussão

Em todos os experimentos apresentados, os subconjuntos selecionados pela estratégia *k-core* obtiveram médias de tempo de execução mais baixas que os obtidas pelos subconjuntos selecionados pela estratégia *ping*. O subconjunto `core-150ms-everyround` foi em média 24% mais rápido que o subconjunto `ping.uk` na aplicação *Triangular.py*, 3% na aplicação *Bit.py* e 28% na aplicação *Torrent*. Comparada ao subconjunto `ping.c3sl`, o subconjunto `core-150ms-everyround` foi em média 61% mais rápido na aplicação *Triangular.py*, 20% mais rápido na aplicação *Bit.py* e quase duas vezes e meia mais rápido na aplicação *Torrent*.

A estratégia *k-core* também foi mais eficiente para minimizar o impacto de fatores externos nos resultados dos experimentos, reduzindo o coeficiente de variação dos resultados obtidos. O subconjunto `core-150ms` obteve um coeficiente de variação de 0,65 na aplicação *Triangular.py*, contra 0,916 do subconjunto `ping.uk` e 0,841 do subconjunto `ping.c3sl`. Os coeficientes obtidos na aplicação *Bit.py* também foram menores,

0,067 do subconjunto `core-150ms` contra 0,092 e 0,093 dos conjuntos `ping.uk` e `ping.c3sl`. O coeficiente de variação obtido pelo subconjunto `core-150ms` é novamente mais baixo na aplicação *Torrent*, 0,155, enquanto `ping.uk` e `ping.c3sl` obtiveram respectivamente 0,191 e 0,171.

Comparando os dois subconjuntos obtidos pela estratégia *k-core*, os resultados obtidos são bastante similares, com médias de tempo de execução equivalentes em todas as aplicações, coeficientes de variação idêntico na aplicação *Bit.py* e bastante próximos nas aplicações *Triangular.py* e *Torrent*. Os resultados obtidos pelo subconjunto `core-150ms` demonstram a estabilidade dos nodos selecionados pela estratégia *k-core*. Durante um período de 40 dias, o subconjunto permaneceu fixo e manteve as propriedades de conectividade entre os nodos, obtendo resultados com baixa variação durante o período de experimentação.

A estratégia de seleção de nodos `core-150ms-everyround` inclui um número maior de nodos no experimento. Como foi observado em um período de três dias, a seleção de 150 nodos executada antes do início de cada uma das 30 rodadas de experimento de uma série incluiu um total de 262 nodos diferentes. É esperado que a utilização de diferentes nodos resulte em uma maior variação dos resultados, no entanto, os resultados obtidos pelo subconjunto `core-150ms-everyround` foram muito similares ao subconjunto fixo `core-150ms`. Portanto, a estratégia de seleção de nodos estáveis é capaz de selecionar subconjuntos de nodos com características de estabilidade semelhantes apenas utilizando os mesmos parâmetros de seleção, contribuindo para a reprodutibilidade dos experimentos.

5. Conclusão

Novas soluções, arquiteturas e serviços para a Internet precisam ser testados em um ambiente realista e de larga escala. O PlanetLab é um *testbed* de escala planetária, com mais de mil nodos espalhados ao redor do mundo, o qual oferece aos pesquisadores um ambiente real para a execução de experimentos. No entanto, a execução de experimentos no PlanetLab pode atingir um alto grau de complexidade, especialmente pela existência de instabilidades e variações de desempenho nos nodos e na própria rede. Este trabalho apresentou uma avaliação experimental do impacto do uso de uma ferramenta de seleção na precisão dos resultados obtidos em experimentos realizados no PlanetLab.

Os resultados obtidos confirmaram que o PlanetLab é um ambiente instável e é muito difícil obter repetibilidade nos experimentos realizados. No entanto, estas variações não atingem todos os nodos da mesma maneira e com a mesma intensidade. A utilização de subconjuntos de nodos obtidos pela estratégia de seleção *k-core* reduziu as variações observadas nos resultados obtidos, aumentando a precisão e contribuindo para a repetibilidade dos experimentos. Além disso, os subconjuntos de nodos selecionados pela ferramenta de monitoração obtiveram melhores médias de tempo de execução para as aplicações testadas, demonstrando a capacidade da estratégia de seleção de encontrar um subconjunto de nodos com melhor conectividade entre todos os pares, confirmando os resultados originais obtidos pelo autor da ferramenta.

Os experimentos realizados neste trabalho demonstraram que a ferramenta de seleção de nodos baseada no monitoramento das interações fim-a-fim é eficiente em obter subconjuntos de nodos estáveis para a realização de experimentos. Porém, existem outras

informações sobre os nodos que a ferramenta de seleção não foi projetada para monitorar. Durante a execução dos experimentos, foi detectada especialmente a falta de informações relacionadas à disponibilidade de memória e espaço em disco dos nodos. Um trabalho futuro importante seria a realização de novos experimentos para avaliar a incorporação da monitoração de outros parâmetros à estratégia de seleção, permitindo uma filtragem de nodos selecionados pela estratégia *k-core*.

Referências

Albrecht, J., Oppenheimer, D., Vahdat, A., and Patterson, D. (2008). Design and implementation trade-offs for wide-area resource discovery. *ACM Transactions on Internet Technology (TOIT)*, 8(4):18.

Barros Neto, B. and Scarminio, I. (2002). *Como Fazer Experimentos: pesquisa e desenvolvimento na ciência e indústria*. Editora da Unicamp.

Bona, L., Duarte Jr., E., and Garrett, T. (2012). Monitoring pairwise interactions to discover stable wormholes in highly unstable networks. In Korakis, T., Zink, M., and Ott, M., editors, *Testbeds and Research Infrastructure. Development of Networks and Communities*, volume 44 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 146–161. Springer Berlin Heidelberg.

Box, G., Hunter, W., and Hunter, J. (2005). *Statistics for experimenters: an introduction to design, data analysis, and model building*. Wiley series in probability and mathematical statistics. Wiley.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. (2003). Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12.

Dean, J. and Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *OSDI 2004*, pages 137–150.

Duarte Jr., E., Garrett, T., Bona, L., Carmo, R., and Zuge, A. (2010). Finding stable cliques of planetlab nodes. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 317–322. IEEE.

Eastlake 3rd, D. (2001). RFC 6234: Secure hash algorithm.

Lenders, V. and Martonosi, M. (2009). Repeatable and realistic experimentation in mobile wireless networks. *Mobile Computing, IEEE Transactions on*, 8(12):1718–1728.

Pan, J., Paul, S., and Jain, R. (2011). A survey of the research on future internet architectures. *Communications Magazine, IEEE*, 49(7):26–36.

Park, K. and Pai, V. S. (2006). Comon: A mostly-scalable monitoring system for planetlab. *SIGOPS Oper. Syst. Rev.*, 40(1):65–74.

Spring, N., Peterson, L., Bavier, A., and Pai, V. (2006). Using planetlab for network research: Myths, realities, and best practices. *SIGOPS Oper. Syst. Rev.*, 40(1):17–24.

Y., W., Carzaniga, A., and Wolf, A. (2008). Four enhancements to automateddistributed system experimentation methods. In *Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on*, pages 491–500.

Bluemob: Algoritmo Dinâmico para Formação de Redes Bluetooth em Aplicações de Sensoriamento Urbano

Giovani Pieri[1], Werner Kraus Jr.[1], and Jean-Marie Farines[1]

[1]Departamento de Controle e Automação – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

pieri@das.ufsc.br, {werner.kraus,j.m.farines}@ufsc.br

Abstract. *We propose a new algorithm for building ad-hoc networks among mobile devices tailored for small and dynamic environments using the Bluetooth interface. The target application is network formation among passengers on urban buses. The proposed algorithm assigns roles to the network participants for reducing the effect of nodes leaving the network upon alighting. Based on estimates of node permanence onboard, an application dependent mechanism assign roles to the nodes indicating those more apt to perform a given function in the network (masters, slaves or gateways). Simmulation studies indicate improvements of up to 75% in the times needed to rebuild the network after nodes have left in comparison to a classic algorithm. Performance levels in relation to the quality of estimates are also evaluated for indication of accuracy of estimation needed for proper application of the algorithm.*

Resumo. *Propomos um algoritmo para formação de redes ad-hoc entre dispositivos móveis em ambientes pequenos e dinâmicos através da interface Bluetooth. A aplicação pretendida é a formação de rede entre passageiros de ônibus urbanos. O algoritmo proposto atribui funções aos integrantes da rede de forma a atenuar o efeito da saída de dispositivos no desembarque. Baseando-se em estimativas de permanência de cada nó a bordo do ônibus, um mecanismo dependente da aplicação atribui papéis aos nós, indicando quais os mais aptos a ocupar uma determinada função na rede (mestre, escravo ou* gateway*). Experimentos de simulação apontam melhora de até 75% nos tempos de reformação após a saída de nós da rede em comparação com algoritmo clássico. Avaliam-se também os níveis de desempenho em relação à qualidade das estimativas, indicando qual a precisão das estimativas necessária para a boa aplicação do algoritmo.*

1. Introdução

Há muitas possibilidade que surgem de redes ad-hoc formadas por smartphones em ambientes fechados. Por exemplo, robôs explorando uma construção podem se comunicar entre si para aperfeiçoar procedimentos de mapeamento; pessoas em um encontro social podem contactar pessoas próximas que estejam dispostas a interagir; e passageiros de ônibus podem, além de interagir entre si através da rede, contribuir para coletar dados sobre o transporte público.

Este trabalho é orientado ao último caso. O objetivo é organizar uma rede que permita o levantamento de informações sobre dados de ônibus para auxílio no planejamento

e operação dos sistemas de transporte público. Tal esquema dispensaria a necessidade de implantação de infraestrutura pública por autoridades e operadores, tanto nos ônibus quanto nas cidades. Duas vantagens da proposta são (1) baixo custo para a cidade e operadores (tanto em infraestrutura quanto em manutenção), importante em situações onde operadores resistem em investir no serviço de transporte público; e (2) a independência e resiliência do sistema de coleta em relação a interesses de agentes que se beneficiariam ocultando informações. Em cenários nos quais estes fatores não são relevantes, o sistema beneficia passageiros interessados em interagir entre si no ônibus através da rede formada.

A delegação da solução de um problema e obtenção de informações sobre o ambiente urbano a voluntários que resulta em um retorno social é conhecido como *urban sensing* [Burke et al. 2006, Campbell et al. 2006]. O sucesso de aplicações como Waze [Waze Mobile] atesta a viabilidade de *urban sensing* usando smartphones para coleta de informações, neste caso sobre o estado do tráfego. Trabalhos anteriores da literatura que fazem uso de smartphones para coleta de dados são: Nericell [Mohan et al. 2008] que usa smartphones para monitorar condições de trânsito e manutenção de rodovias através dos sensores do smartphones para automaticamente detectar frenagens, acelerações, lombadas, buracos e níveis de ruído; [Thiagarajan et al. 2010] descreve um sistema de rastreamento de ônibus e trem para estimar o tempo de chegada do próximo ônibus ou trem na parada; em [Lin et al. 2010] um sistema é proposto para avaliar o nível de conforto do transporte público através de dados dos sensores dos smartphones.

Há trabalhos na literatura acerca da obtenção de dados sobre passageiros em sistemas de transporte público. Por exemplo, [Ben Moshe et al. 2014] propõe o uso de equipamento no interior do ônibus para detectar e identificar dispositivos Bluetooth dos passageiros. Entretanto, equipamentos devem ser instalados no interior do ônibus e smartphones assumem um papel passivo. Baseado no sucesso de *Waze*, *Moovit* [Moovit] provê informação obtida junto aos usuários e operadores, porém usa interface de dados (3G ou similar) com alto consumo energético e não coleta informação sobre viagens realizadas.

Nossa abordagem se apoia na formação de redes espontâneas no interior do ônibus pelos seguintes motivos. Primeiro, é possível manter um registro da vida da rede desde o início da viagem do ônibus até o seu destino sempre que não houver desembarque total em algum ponto ao longo do percurso. Segundo, a transmissão de informações a um servidor para processamento de dados pode ser realizada por um único nó, possivelmente um líder escolhido na rede, economizando tráfego e ajudando na consistência de dados. Terceiro, apesar deste aspecto não ser tratado neste trabalho, é mais fácil confirmar que smartphones estão em um mesmo ônibus quando eles podem trocar informações entre eles (p. ex., através da troca de traços de GPS). O problema de determinar quando um smartphone está de fato embarcado em um ônibus foi abordado anteriormente e mostrado ser de difícil tratamento [Reddy et al. 2010].

Bluetooth é usado como interface de comunicação pois (1) é mais eficiente energeticamente do que WiFi; (2) ônibus são particularmente pequenos (entre 12 m e 18 m) e o maior alcance fornecido pelo WiFi não é necessário para construir uma rede entre passageiros de um ônibus; e (3) Bluetooth tem suporte para criar redes sem infraestrutura.

Considerando que a abordagem se apoia em redes Bluetooth, a questão da formação de tais redes é tratada no restante deste trabalho.

A principal restrição para o algoritmo de formação de redes é temporal: o tempo para formação da rede deve ser menor que o tempo de viagem entre dois pontos de paradas. Tempos típicos entre paradas são de 1 min, apesar de haver tempos menores em situações com paradas pouco espaçadas. P. ex., considerando-se aceleração, velocidade de cruzeiro e frenagem, velocidades médias de 30 km/h (8.3 m/s) são comumente observadas, resultando em intervalos de 30 s para paradas espaçadas de 250 m.

Neste artigo, propomos *Bluemob*, um algoritmo projetado para esta aplicação. Baseado em algoritmos de formação previamente propostos na literatura, adicionam-se pesos relativos à estimativa de permanência a bordo de forma a posicionar nas bordas da rede aqueles nós com maior propensão a desembarcar. Desta forma, busca-se evitar o particionamento da rede durante os desembarques, provendo a reconstrução da rede de forma mais rápida. Resultados de simulação indicam que os tempos envolvidos para a reformação da rede são menores do que o tempo típico de viagem entre paradas.

A topologia resultante de Bluemob reflete a propensão dos nós em saírem da rede. Salientamos que este mesmo conceito pode ser usado em cenários com restrições diferentes das temporais. P. ex., a energia dos nós pode ser mais importante na comunicação multi-robôs e ambientes sociais. Nestes casos, Bluemob deve ser adaptado para que a topologia da rede reflita estimativas energéticas tais como a carga e capacidade da bateria, e consumo previsto considerando o histórico do uso de cada dispositivo.

O artigo é organizado da forma seguinte. Na seção 2 são apresentados algoritmos para construção de redes Bluetooth existentes na literatura. Na seção 3 apresentamos os detalhes de Bluemob, destacando suas propriedades. A avaliação de Bluemob e resultados de simulação são apresentados na seção 4. Por fim, na seção 5 retomamos as principais contribuições do artigo e apontamos perspectivas futuras.

2. Algoritmos para construção de redes Bluetooth

Bluetooth é uma tecnologia de comunicação para curtas distâncias. *Smartphones* e outros dispositivos de uso pessoal o utilizam devido a seu baixo custo e consumo energético. O protocolo Bluetooth é baseado em comunicação mestre-escravos e mecanismo de salto de frequência na faixa de 2.4 GHz. São usados 79 canais com 1 MHz de diferença entre eles.

O processo de estabelecimento de uma conexão Bluetooth se dá através do processo INQUIRY e INQUIRY SCAN. Um dispositivo realiza buscas através de INQUIRY, enviando mensagens em uma sequência de 32 canais pré-definidos. Dispositivos usam o processo INQUIRY SCAN para serem descobertos, no qual ouvem mensagens enviadas nos mesmos 32 canais usados pelo INQUIRY. Ao receber um pacote enviado por INQUIRY, o dispositivo INQUIRY SCAN estabelece uma conexão. O dispositivo em INQUIRY se torna o mestre do dispositivo em INQUIRY SCAN.

Um mestre pode se conectar a no máximo sete escravos ativos. Esta estrutura é denominada *Piconet*. Toda a comunicação se dá entre o escravo e seu mestre. Não há comunicação direta entre dois mestres ou dois escravos. Dispositivos podem participar de mais de uma piconet ao mesmo tempo, denominados pontes. Escravos que participam de apenas uma piconet são denominados escravos não compartilhados. Redes formadas por piconets interconectadas são chamadas de *Scatternets*.

A formação de *scatternets* (ou *Bluetooth Scatternet Formation* - BSF) consiste em

definir como as piconets são formadas a partir de um conjunto de dispositivos Bluetooth que se conhecem entre si e como estas são topologicamente organizadas [Jedda et al. 2013]. Os algoritmos BSF devem atribuir o papel de mestre, escravo ou ponte a cada um dos participantes da rede e assegurar a conectividade desta, respeitando ainda as restrições impostas pela tecnologia Bluetooth, como o limite de sete escravos por mestre.

Os algoritmos BSF tentam resolver o problema de formação da *scatternet* otimizando diferentes métricas. Dentre as métricas, destacam-se: o tempo necessário para formar a *scatternet*; o número de *piconets*; quantidade de pontes mestre-escravo (ME) e escravo-escravo (EE), priorizando pontes EE sobre pontes ME; e número de papéis por nó, ou seja, a quantas piconets cada nó pertence em média [Jedda et al. 2013].

Os algoritmos BSF podem ser classificados em [Stojmenovic and Zaguia 2006]:

- Algoritmos *Single-hop* vs. *multi-hop*: algoritmos *single-hop* pressupõem que todos os dispositivos encontram-se dentro do alcance de comunicação Bluetooth, permitindo que quaisquer dois destes se conectem. Os algoritmos *Multi-hop* correspondem ao caso da existência de dispositivos fora do alcance de comunicação um do outro, necessitando o roteamento de mensagens.
- Algoritmos centralizados vs distribuídos: no caso centralizado, o algoritmo BSF é executado numa unidade central de processamento, como um servidor externo ou um líder. BTCP [Salonidis et al. 2005] é um exemplo de algoritmo desta classe. No caso distribuído, o processamento é local, dados são trocados entre vizinhos e a decisão da topologia final da rede é realizada de forma distribuída.
- Descoberta dinâmica vs descoberta estática: em algoritmos com descoberta estática, existe uma fase preliminar de descoberta responsável por identificar todos os dispositivos Bluetooth próximos. BlueStar [Petrioli et al. 2003], BlueMesh [Petrioli et al. 2004], BlueNet [Wang et al. 2002], BlueMIS [Zaguia et al. 2008] são algoritmos com descoberta estática. Os algoritmos com descoberta dinâmica não possuem uma etapa preliminar de descoberta, sendo este procedimento realizado junto ao algoritmo de formação. Portanto, algoritmo com descoberta dinâmica são melhores adaptados a cenários onde o conjunto de dispositivos não é conhecido a priori, ou quando novos dispositivos surgem durante o processo de formação de scatternet.

Um algoritmo BSF para nossa aplicação deve possuir as seguintes características:

- *Single-hop*: nos cenários considerados, o alcance máximo dos dispositivos Bluetooth, é suficiente para cobrir toda a área onde a rede opera. Dispositivos modernos possuem rádios Bluetooth Classe 2 capazes de estabelecer canais de comunicação a distâncias de até 20 metros [Pei et al. 2010], suficientes para cobrir todo o ônibus, dispensando o uso de algoritmos *multi-hop* e sua complexidade.
- Descoberta Dinâmica: nos cenários considerados de redes formadas entre passageiros de ônibus, há entrada e saída constante de nós da rede, correspondente aos embarques e desembarques do ônibus. Consequentemente, um algoritmo BSF deve ser capaz de incorporar novos dispositivos a *scatternet*. Este cenário é mais favorável a algoritmos com descoberta dinâmica.

Os algoritmos Law [Law et al. 2003] e TSF [Tan et al. 2002] são os dois principais algoritmos BSF *single-hop* com descoberta dinâmica.

Law e TSF constroem componentes, e expandem a scatternet através da união (*merge*) destes. As principais diferenças entre Law e TSF são: (i) a organização dos componentes numa árvore para o algoritmo TSF, enquanto o algoritmo Law cria redes *mesh*; (ii) pequenas diferenças no processo de busca por novos nós, apesar da similaridade entre os dois.

Neste trabalho escolhemos o algoritmo Law como ponto de partida para a proposição de um novo algoritmo BSF devido ao uso da topologia *mesh*. Isso permite maior liberdade durante a reorganização da scatternet que ocorre durante a entrada e saída de nós da rede, sendo mais apropriada ao nosso objetivo. Entretanto, o algoritmo TSF pode ser adaptado para operar nestes cenários, mas com um *overhead* extra introduzido pela obrigatoriedade da manutenção da topologia em árvore.

3. Algoritmo BSF Dinâmico para Aplicações de Sensoriamento Urbano

Algoritmos BSF dinâmicos propostos na literatura não levam em consideração que alguns nós da rede têm maior propensão a deixar a *scatternet*. Esta situação é intrínseca do ambiente no qual Bluemob opera. Atribui-se a cada nó da rede um número, chamado *peso*, para quantificar a sua propensão em desembarcar. A introdução de um modelo de comportamento na formação possibilita alocar nós com menor propensão de sair da rede em posições de mestre e ponte. Desta forma, durante a saída dos nós da rede haverá um menor número de particionamentos em relação a algoritmos sem modelo de comportamento pois os nós que saem da rede não ocuparão posições estruturantes. Menos particionamentos acarretam tempo menor para restaurar a conectividade da rede.

Possíveis indicadores da propensão de um passageiro para desembarcar são, por exemplo: (i) o tempo que o passageiro está a bordo, considerando que quanto maior este tempo, mais próximo está o seu ponto de desembarque; (ii) a posição do passageiro no interior do ônibus, pois a proximidade dos passageiros às portas de saída indica uma propensão a desembarcar, principalmente em ônibus lotados; (iii) o conhecimento prévio sobre padrões de embarques e desembarques em uma dada rota de ônibus, obtido a partir de dados históricos de origem-destino que fornecem a probabilidade de desembarque em uma parada dado o local onde o embarque ocorreu. Neste trabalho, o modelo de comportamento é traduzido em um número que indica o ponto de ônibus de saída do nó. Embora o processo de obtenção da estimativa não seja discutido, variam-se os níveis de precisão da estimação para avaliar o quão crítico é este processo.

Bluemob é um algoritmo BSF com descoberta dinâmica inédito, diferenciando-se por considerar a aplicação na estruturação da rede através do peso a ele atribuído. A melhoria no desempenho do algoritmo pode ser medida pelo tempo que necessita para reestabelecer a rede após a saída de nós. No cenário em estudo, a rede deve ser sempre reconstruída antes da chegada do ônibus na próxima parada. O tempo de viagem entre paradas, então, impõe um *soft deadline* que precisa ser respeitado para que as aplicações possam identificar os passageiros embarcados que desejam participar da rede.

O peso h do nó i utilizado no algoritmo Bluemob pode assumir quaisquer valores, desde que exista uma relação de ordem total definida sobre os valores possíveis de $h(i)$.

Um componente é definido como sendo: (1) um nó desconectado; (2) uma piconet; ou (3) uma *scatternet*. Todo componente possui um *líder* responsável por coordenar as

atividades de buscas por dispositivos. Este papel é desempenhado por um de seus mestres. Caso o componente possua um único nó, então este nó será o líder do componente. Um componente válido deve satisfazer as seguintes propriedades:

(i) o componente possui um único líder;
(ii) o líder deve possuir no máximo seis escravos;
(iii) os mestres devem possuir menos que sete escravos;
(iv) o líder deve possuir um escravo não-compartilhado, ou não possuir nenhum escravo;
(v) todo escravo não-compartilhado possui peso h menor ou igual ao peso h de seu mestre.

No que segue será utilizada a seguinte notação:

- $S(x)$: o conjunto de escravos de x;
- $M(x)$: o conjunto de mestres de x;
- $S_u(x)$: o conjunto de escravos não compartilhados de x;
- $Y_p(u)$: o conjunto de escravos não compartilhados de u que têm h maior que ou igual a todos os outros escravos não-compartilhados de u, i. é: $Y_p(u) \leftarrow \{y \in S_u(u) | \forall k \in S_u(u) : h(y) \geq h(k)\}$

O processo básico de Bluemob é a união de componentes. Este processo transforma dois componentes em um componente válido, ou seja, as propriedades acima são verificadas no novo componente gerado.

O algoritmo Bluemob opera em rodadas, sendo que cada rodada possui uma duração δ durante a qual o líder tenta realizar a união com outro componente. No início de uma rodada (linha 3 do algoritmo 1) o líder realiza um processo INQUIRY (linha 6) ou requisita a um de seus escravos não-compartilhados que realize INQUIRY SCAN (linha 12). A escolha entre qual dos dois processos (INQUIRY ou INQUIRY SCAN) será realizado é aleatória com probabilidade P de realizar INQUIRY (linha 5). Caso o componente consista de um único nó, o líder realiza o processo INQUIRY SCAN ele mesmo (linha 9). É importante notar que este algoritmo pressupõe que todas as operações da rodada, incluindo o processo de união, sejam concluídas dentro do limite de tempo δ.

Ao estabelecer uma conexão, dados sobre os componentes são trocados e o processo de reorganização da topologia iniciado.

Seja $h(u)$ o valor de h do nó u, e $y(u) \in Y_p(u)$. O mestre em INQUIRY é u; o nó em INQUIRY SCAN é v; e o líder do componente de v é denominado w. O conjunto dos escravos de um nó qualquer x é denominado $S(x)$. Salienta-se que, se v é um escravo de u ($v \in S(u)$) e o componente de v possui mais do que um nó ($v \neq w$), então v também será escravo do líder w ($v \in S(w)$). Em outras palavras, quando o líder u encontra um outro componente, uma conexão é estabelecida entre v e u sendo que u assume o papel de mestre de v. Também, por construção, se v não está sozinho em seu componente, então w será mestre de v. Isto é importante, pois v pertence tanto a $S(w)$ quanto a $S(u)$. Este fato deve ser levado em consideração durante o processo de reorganização.

Quando o líder em INQUIRY descobre outro componente, uma reorganização é feita de forma que o componente resultante respeite as propriedades estabelecidas. Bluemob escolhe uma dentre treze reorganizações, dependendo da estrutura dos componentes:

1. Se $v = w$, o componente descoberto é composto por um nó. Pois, um líder w só entra em INQUIRY SCAN neste caso. Para este cenário, temos cinco casos:

Algorithm 1 Bluemob executando no nó i

```
1: P ← (0, 1); δ ← duração da rodada
2: isLeader_i ← true; h_i ← h(i)
3: procedure START ROUND
4:     p ← random number between 0 and 1
5:     if p < P then
6:         INQUIRY
7:     else
8:         if S(i) = ∅ then
9:             INQUIRYSCAN(i)
10:        else
11:            y ← any element ∈ Y_p(i)
12:            INQUIRYSCAN(y)
13:        end if
14:    end if
15: end procedure
16: procedure WHEN CONNECT(to)
17:    u ← i; v ← to
18:    if M(v) = {u} then
19:        w ← v
20:    else
21:        w ← M(v) − {u}
22:    end if
23:    if Y_p(i) = ∅ then
24:        y ←⊥
25:    else
26:        y ← qualquer elemento ∈ Y_p(i)
27:    end if
28:    REORGANIZE(u, v, w, y)
29: end procedure
30: procedure BECOME LEADER(u)
31:    leader_u ← true
32:    Agendar a execução da tarefa periódica de Start Round com período δ
33: end procedure
34: procedure RETIRE(u)
35:    leader_u ← false
36:    Parar a execução da tarefa periódica de Start Round com período δ
37: end procedure
```

(a) Caso 1: $|S(u)| < 7$ e $h(v) \leq h(u)$
(b) Caso 2: $|S(u)| < 7$ e $h(v) > h(u)$
(c) Caso 3: $|S(u)| = 7$ e $h(u) \geq h(v)$ e $h(y) \leq h(v)$
(d) Caso 4: $|S(u)| = 7$ e $h(u) \geq h(v)$ e $h(y) > h(v)$
(e) Caso 5: $|S(u)| = 7$ e $h(u) < h(v)$

2. Se $v \neq w$, o componente descoberto é uma piconet ou scatternet. Neste cenário, temos os seguintes casos:
   (a) Caso 6: $|S(u)| + |S(w)| < 7$ e $h(u) \geq h(w)$
   (b) Caso 7: $|S(u)| + |S(w)| < 7$ e $h(u) < h(w)$
   (c) Caso 8: $|S(u)| = 1$ e $h(v) \geq h(u)$
   (d) Caso 9: $|S(u)| = 1$ e $h(v) < h(u)$
   (e) Caso 10: $|S(u)| + |S(w)| == 7$ e $h(v) \geq h(w)$
   (f) Caso 11: $|S(u)| + |S(w)| == 7$ e $h(v) < h(w)$
   (g) Caso 12: $h(w) \geq h(u)$
   (h) Caso 13: $h(w) < h(u)$

Para cada caso, Bluemob define os papéis dos nós das piconets dos líderes (u e w), refazendo conexões e determinando quem será o líder do novo componente.

Devido a restrições de espaço, não é possível detalhar as treze reorganizações. A descrição completa do algoritmo e suas reorganização podem ser encontradas no relatório

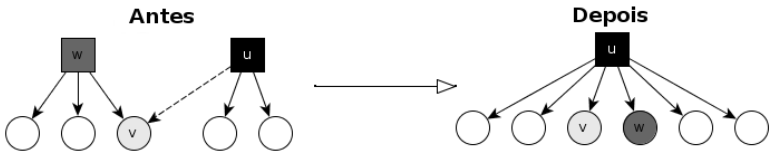


Figura 1. Procedimento de reorganização após união de componentes (Caso 7).

técnico [Pieri 2014]. Para ilustrar, a Figura 1 mostra o caso 7, no qual u realiza o processo de INQUIRY, descobre o nó v e estabelece a conexão entre u e v representada pela linha tracejada. O mestre de v é w, líder do componente. Neste exemplo, o peso de u é menor que o de w ($h(u) < h(w)$). Dado o número de escravos na piconet de cada um dos líderes, $|S(u)| + |S(w)| < 7$, os nós de ambas as piconets podem ser unificadas em uma única piconet sem violar o limite superior de escravos do líder (neste exemplo, seis). A questão pendente em relação à reorganização é a seguinte: deve u se tornar escravo de w e w se tornar o novo líder, ou deve w se tornar escravo de u e u assumir o papel de novo líder?

Para que o novo componente seja válido, as propriedades devem ser respeitadas após a reorganização. Como $h(u) < h(w)$, u não pode ser mestre de w, então, u se torna escravo de w. Os escravos de u tornam-se escravos de w pois a terceira propriedade diz que todo escravo não compartilhado deve possuir h menor ou igual ao h de seu mestre. Logo, todos os escravos não-compartilhados de u possuem um h menor que u, e, por transitividade, menor que w. Consequentemente, o algoritmo pode concluir que w é o novo líder, e todos os escravos de u se tornam escravos de w sem que nenhuma propriedade seja violada, conforme ilustrado na figura 1.

Os demais casos seguem processos similares para gerar componentes válidos. Dois resultados garantem a preservação das propriedades por Bluemob e, consequentemente, sua vivacidade e corretude, conforme segue.

Lema. *Dado que dois componentes A e B respeitam as propriedades antes da união, segue que: (1) o componente resultante possui um único líder; (2) o líder do componente resultante possui no máximo seis escravos; (3) não existe nenhum mestre com mais de sete escravos no componente resultante; (4) o líder do componente resultante possui pelo menos um escravo não-compartilhado; (5) todos os escravos do componente resultante que não são pontes possuem h menor ou igual ao h de seu mestre.*

Teorema. *Todo componente respeita as seguintes propriedades: (1) o componente possui um único líder; (2) o líder deve possuir no máximo seis escravos; (3) os mestres devem possuir menos que sete escravos; (4) o líder deve possuir um escravo não-compartilhado ou nenhum escravo; (5) todo escravo que não é uma ponte possui h menor ou igual ao h de seu mestre.*

As provas do lema e do teorema podem ser encontradas no relatório técnico [Pieri 2014].

4. Avaliação de Bluemob

O algoritmo Bluemob deve ter desempenho suficiente para reconstruir a rede de nós a bordo de um ônibus após o desembarque de passageiros num tempo menor que o tempo

típico de viagem entre pontos de paradas. Também, pode-se avaliar a melhoria de Bluemob em relação ao algoritmo de Law usado como base. Esta seção realiza tal avaliação usando como métrica comparativa o número de rodadas para formação e reformação da rede. São avaliados os efeitos de (1) iniciar a formação de uma rede a partir de componentes individuais; (2) passageiros deixando a rede e possivelmente particionando-a; e (3) precisão das estimativas do peso h.

Os experimentos usam um simulador especialmente desenvolvido em Python. Este simulador não leva em consideração detalhes da pilha Bluetooth. Isto é possível pois os algoritmos simulados baseiam-se em tarefa periódica (ou *rodada*) de período igual a δ. Logo, o tempo para a união de todos os componentes (i. é, para obtenção de conectividade total da rede) é δ multiplicado pelo número de rodadas.

Naturalmente, é preciso conhecer o valor de δ para concluir algo sobre os tempos envolvidos na formação da rede. O limite inferior de δ é o tempo necessário para concluir a união de quaisquer dois componentes desde o INQUIRY ou INQUIRY SCAN do líder.

Através de experimentos no simulador UCBT [Wang and Agrawal], principal simulador Bluetooth de código aberto, observou-se que o maior tempo para reorganizar dois componentes foi de 7.75 s. Este tempo foi medido através da simulação de todos os treze casos possíveis listados na Seção 3, variando o tamanho das piconets e os pesos h de u, w, y, v de tal forma que todas as ordens relativas entre os pesos fossem contempladas.

Nota-se que o tempo da rodada do algoritmo de Law é pelo menos da mesma magnitude daqueles verificados para Bluemob. Isto porque a reorganização mais demorada (número 6) de Bluemob cujo tempo máximo é 7.75 s é idêntica a uma das reorganizações usadas no algoritmo Law. Logo, δ de Law tem limite inferior igual a 7.75 s. Deste ponto em diante, escolhemos que $\delta = 8$ s.

4.1. Primeiro experimento: formação a partir de uma rede desconectada

Neste experimento é avaliado o comportamento de Bluemob para formar uma rede a partir de nós desconectados. Este cenário equivale à formação da rede no ônibus entre seu ponto de origem até o primeiro ponto de parada. A simulação é executada até ser formada uma scatternet conectada com todos os nós. O número de rodadas necessárias para formar a *scatternet* é mostrado na Figura 2) para quantidades de nós variando entre 2 e 128.

Observa-se que o tempo para formação da rede está entre $\delta \times 10 = 80$ s e $\delta \times 11 = 88$ s com 128 passageiros. Dado que o tempo típico entre paradas é 60 s, com 128 passageiros não se consegue formar a rede até a próxima parada. Entretanto, esta situação não é usual durante a trajetória do ônibus, sendo observada normalmente apenas na origem do itinerário. Neste ponto, os desembarques não costumam ocorrer durante as primeiras paradas, permitindo ultrapassagem da restrição.

O número de rodadas aumenta logaritmicamente em relação ao número de componentes, não havendo aumentos expressivos no tempo de formação observado com o aumento de passageiros.

Em comparação ao algoritmo Law, pode-se ver na Figura 2 que não há diferenças em relação ao número de rodadas necessárias para a formação da rede. Isto é, não há penalidade devido à maior complexidade da união de componentes de Bluemob. Este comportamento pode ser explicado pela forma como a busca por componentes é realizada

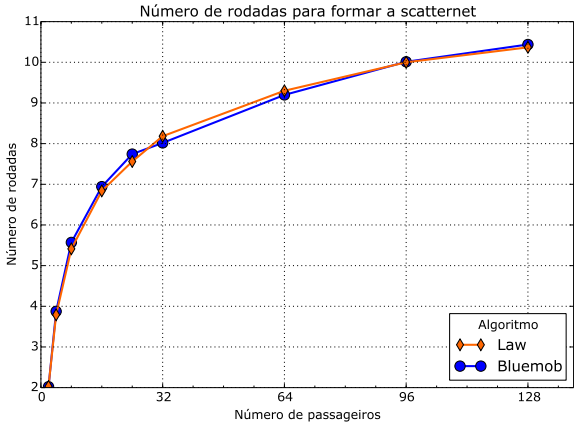
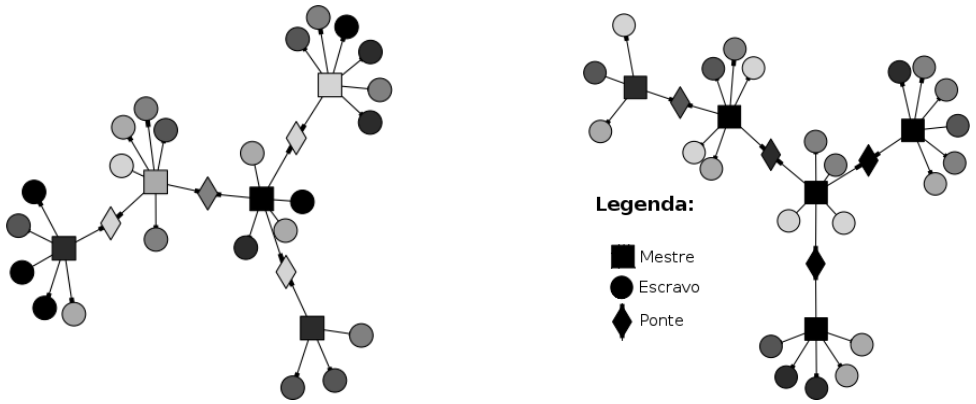


Figura 2. Número de rodadas para formação de rede a partir de uma rede desconectada, quantidade de nós variando de 0 a 128



(a) Law et. al (b) Bluemob

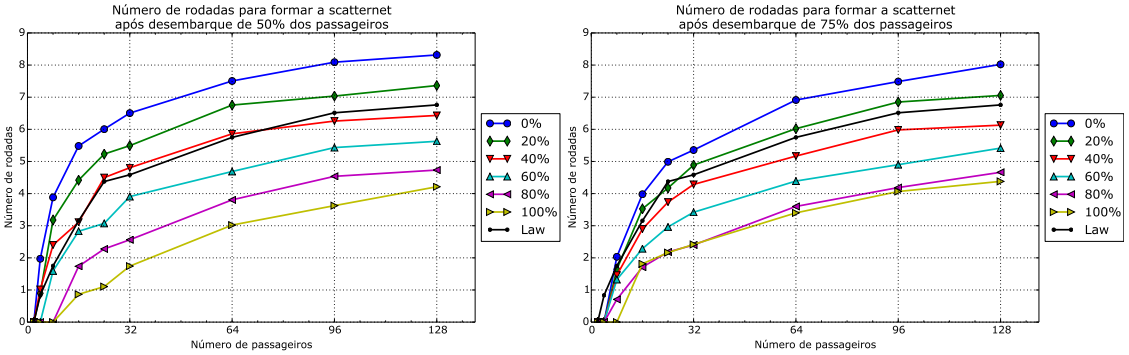
Figura 3. Uma rede Bluetooth formada pelo (a) algoritmo Law, e (b) algoritmo Bluemob; quanto mais clara a tonalidade do nó, maior sua propensão a deixar a rede

em rodadas [Law et al. 2003].

Do ponto de vista da topologia da rede formada, há uma melhora qualitativa no posicionamento dos nós, conforme mostrado na Figura 3, na qual os nós com tonalidade mais clara apresentam maior propensão a deixar a rede. Na rede formada com Bluemob (Figura 3(b)) mestres e pontes tem tonalidade mais escura que os escravos. Logo, quando um nó deixa a rede nesta topologia, é geralmente um escravo e não ocorrem particionamentos. Já na rede formada pelo algoritmo Law (Figura 3(a)) vários mestres e pontes possuem tonalidade clara, indicando maior chance de particionamento da rede por conta de desembarques.

4.2. Segundo experimento: tempo de formação após desembarque de nós

O objetivo do segundo experimento é avaliar o efeito do arranjo de nós quando passageiros deixam a rede, mostrando as melhoras obtidas com a introdução do peso h no procedimento de reorganização durante o desembarque. Este experimento consiste em, dado n nós no ponto 0 do itinerários, construir uma scatternet. No momento em que a scatternet se torna conectada, uma fração dos nós é removida da rede, particionando-a para simular



(a) saída de 50% dos nós da rede (b) saída de 75% dos nós da rede

Figura 4. Experimento reconstruindo scatternet após desembarque

desembarques no ponto 1. A partir deste instante, mede-se o número de rodadas para que a rede seja reconectada no trajeto entre pontos 1 e 2.

O método de atribuição de pesos h é o seguinte: aos nós que deixarão a rede é atribuído o valor 1 a h, enquanto aos nós que permanecerão é atribuído valor 2 a h. A precisão de $x\%$ indica a porcentagem de passageiros com h correto. Assim, $100\% - x\%$ dos nós que deixam a rede tem seu peso h modificado de 1 para 2, enquanto $100\% - x\%$ dos que permanecem na rede tem seu peso h modificado de 2 para 1.

Os experimentos foram simulados 400 vezes para cada uma das situações. O número médio de rodadas para reconectar a rede é mostrado na Figura 4. Dois experimentos foram feitos: com 50%, Fig. 4(a), e 75%, Fig. 4(b), dos nós deixando a rede. Pelas figuras, observa-se novamente o aumento logarítmico do número de rodadas com o número de nós. Em relação à precisão das estimativas, fica claro que quanto melhor for estimada a saída, melhor Bluemob se comporta. Nota-se que no pior caso, com precisão 0% e 128 passageiros, o tempo para reconstrução da rede ultrapassa $60s$. Com o aumento da precisão de h obtém-se tempos na faixa $30s$, atingindo o objetivo de reconstrução da rede entre paradas. Observa-se que a partir do momento em que precisão de h ultrapassa 50%, Bluemob supera o desempenho de Law. Em alguns casos observa-se melhora de até 75% do tempo necessário para reconstruir a rede após saída de parte dos nós.

Quando h possui precisão menor que 50%, Bluemob coloca como mestres e pontes passageiros com alta propensão de desembarque. Com isso, há uma perda de desempenho em relação a Law, que posiciona os nós de forma aleatória na rede.

4.3. Terceiro experimento: tempos de formação durante trajetória

Este experimento tem como objetivo mostrar o comportamento do algoritmo ao longo da trajetória de um ônibus com uma série de embarques e desembarques. Para tanto, assume-se uma trajetória do ônibus conforme mostrado na Fig. 5. A trajetória do ônibus ocorre ao longo de 17 paradas, numeradas sequencialmente sendo a primeira parada a de número 0. Na parada zero ocorre o embarque de 64 passageiros. Nas paradas ímpares, há o desembarque de 16 passageiros e nas paradas pares embarcam novos 16. Os passageiros que desembarcam são aqueles embarcados a mais tempo, e o peso h é igual ao número do ponto estimado para desembarque. A precisão de h indica a porcentagem de passageiros com h correto.

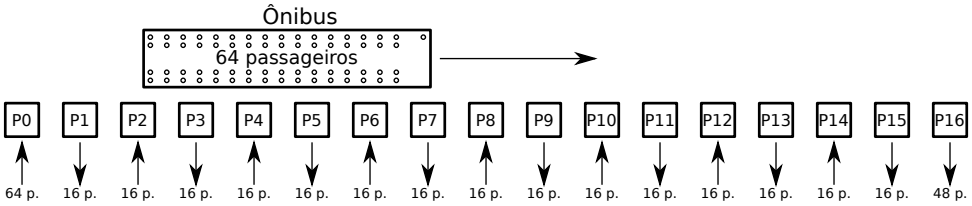


Figura 5. Trajetória de ônibus para teste de reformação da rede em itinerário

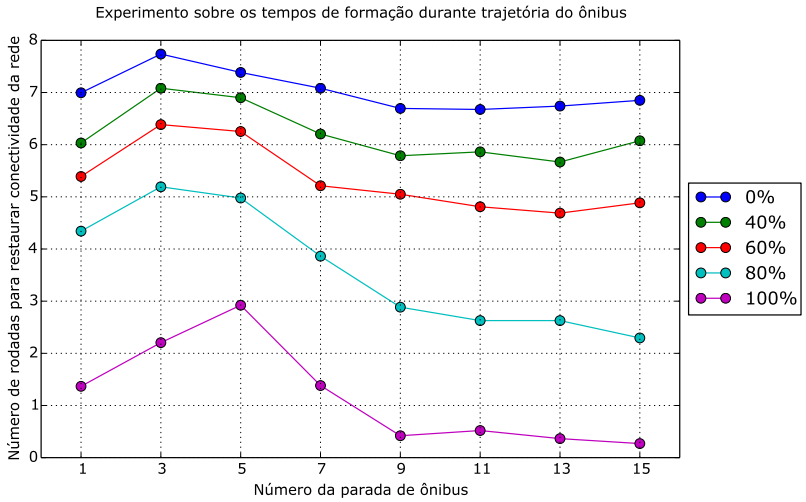


Figura 6. Efeito da saída de passageiros durante trajetória com 64 passageiros 16 desembarques seguidos de 16 embarques alternadamente

Para Bluemob, o importante para definir a corretude de h é a corretude das relações de ordem entre os pesos h. Portanto, o h do passageiro que desembarca em um ponto de parada está correto caso seja menor que o pesos dos passageiros que irão desembarcar posteriormente. Logo, atribui-se aos passageiros com h errados o número de parada menos o número da sua parada de embarque. Por exemplo, passageiros com h errado que embarcaram no ponto de parada 0 tem $h = 17$. Já passageiros que embarcam em 2, terão $h = 17 - 2 = 15$

Na figura 6 é mostrado o efeito da saída de nós da rede quando há uma sequência de entradas e saídas de nós na rede e o efeito em Bluemob. Observa-se que neste experimento os tempos para as reorganizações diminuem durante a trajetória do ônibus. Isto ocorre devido ao modo de operação de Bluemob. Há uma tendência a formação de subredes compostas por passageiros antigos já que dois componentes se unem a partir das piconets dos líderes. Caso haja poucos particionamentos da rede, aglomerados de passageiros antigos são formados. Ao desembarcarem, estes aglomerados de passageiros antigos tendem a desembarcar todos juntos, diminuindo assim o número de partições causadas pelo desembarque.

Após os desembarques iniciais, com os passageiros mais antigos desembarcando primeiro, há uma tendência a estabilização do número de rodadas para reestabelecer a conectividade. Após quatro rodadas de desembarques, praticamente não há necessidade de reorganizações para reconectar a rede quando precisão de h é 100%.

5. Conclusão e Perspectivas

Este trabalho apresenta um novo algoritmo BSF, denominado Bluemob, especificamente projetado para construir redes Bluetooh ad-hoc entre passageiros no interior de ônibus.

O algoritmo proposto melhora algoritmos encontrados na literatura levando em consideração a propensão de um passageiro a deixar a rede, usando esta informação para influenciar a topologia da rede. Passageiros propensos a deixar a rede são posicionados nas bordas da scatternet de tal forma que o seu desembarque não cause o particionamento da rede.

Resultados experimentais obtidos através de simulações são apresentados. Estas simulações mostram que Bluemob melhora algoritmos existentes, desde que provido com um mecanismo para estimar a propensão de desembarques suficientemente preciso. A distribuição dos nós na rede resultante é melhorado, o que é desejável em uma rede formada por passageiros em transporte público. Isto acarreta melhora no tempo necessário para reconstrução da rede após desembarques. Mais importante, o tempo para construir redes encontram-se dentro da faixa aceitável de menos de 1 minuto, tempo típico para viagem de um ônibus entre paradas.

Como trabalhos futuros e área de melhora, pode-se citar os seguintes aspectos: (1) aperfeiçoar o algoritmo para que a topologia evolua para uma distribuição ótima de nós dentro da scatternet, (2) investigar e avaliar métodos para atribuição e comparação da propensão de permanência no ônibus; (3) investigar outras áreas de aplicação, como cenários com restrições energéticas nos quais nós com menor energia possam ser colocados nas bordas da scatternet; (4) desenvolver estudos e comparativos usando dispositivos reais.

Agradecimentos

Os autores agradecem o apoio do CNPq e do projeto SPACeS (Sistema PArticipativo de Gestão de Cidades e Serviços Públicos) do CTIC-RNP.

Referências

Ben Moshe, B., Hadas, Y., and Levi, H. (2014). Energy-Efficient Framework for Indoor and Outdoor Tracking of Public Transit Passengers Using Bluetooth-Enabled Devices. In *Transportation Research Board 93rd Annual Meeting*, pp. 16–30, Washington DC.

Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., and Srivastava, M. B. (2006). Participatory sensing. In *Proc. Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications*, pp. 117–134.

Campbell, A. T., Eisenman, S. B., Lane, N. D., Miluzzo, E., and Peterson, R. A. (2006). People-centric urban sensing. In *Proc. 2nd Annual International Wireless Internet Conference (WICON '06)*, pp. 2–5, New York, NY, USA. ACM.

Jedda, A., Casteigts, A., Jourdan, G.-V., and Mouftah, H. T. (2013). Bluetooth scatternet formation from a time-efficiency perspective. *Wireless Networks*, 20(5):1133–1156.

Law, C., Mehta, A. K., and Siu, K. Y. (2003). A New Bluetooth Scatternet Formation Protocol. *Mobile Networks and Applications*, 8:485–498.

Lin, C.-Y., Chen, L.-J., Chen, Y.-Y., and Lee, W.-C. (2010). A comfort measuring system for public transportation systems using participatory phone sensing. In *Proc. of First Int. Workshop on Sensing for App Phones (PhoneSense'10)*, Zurich, Switzerland.

Mohan, P., Padmanabhan, V. N., and Ramjee, R. (2008). Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proc. 6th ACM Conf. on Embedded Networked Sensor Systems (SenSys '08)*, pp. 323–336, New York, NY, USA.

Moovit. Moovit. `http://www.moovitapp.com/`. Acessado em: 2012-12-12.

Pei, L., Chen, R., Liu, J., Kuusniemi, H., Tenhunen, T., and Chen, Y. (2010). Using inquiry-based bluetooth rssi probability distributions for indoor positioning. *Journal of Global Positioning Systems*, 9(2):122–130.

Petrioli, C., Basagni, S., and Chlamtac, I. (2003). Configuring bluestars: multihop scatternet formation for bluetooth networks. *IEEE Trans. on Computers*, 52(6):779–790.

Petrioli, C., Basagni, S., and Chlamtac, I. (2004). BlueMesh: Degree-Constrained Multi-Hop Scatternet Formation for Bluetooth Networks. *Mobile Netw. and Applications*, 9:33–47.

Pieri, G. (2014). Bluemob: Description of a BSF Algorithm for Building Bus Riders Bluetooth ad-hoc networks. Technical report, Departamento de Controle e Automação, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina. `http://www.das.ufsc.br/~werner/workInProgress/bluemob/TR-algorithmDescription.pdf`.

Reddy, S., Mun, M., Burke, J., Estrin, D., Hansen, M., and Srivastava, M. (2010). Using mobile phones to determine transportation modes. *ACM Trans. Sen. Netw.*, 6(2):13:1–13:27.

Salonidis, T., Bhagwat, P., Tassiulas, L., and LaMaire, R. (2005). Distributed topology construction of Bluetooth wireless personal area networks. *IEEE Journal on Selected Areas in Communications*, 23:633–643.

Stojmenovic, I. and Zaguia, N. (2006). Bluetooth scatternet formation in ad hoc wireless networks. *Performance Modeling and Analysis of Bluetooth Networks*, pp. 147–171.

Tan, G., Miu, A., Guttag, J., and Balakrishnan, H. (2002). An efficient scatternet formation algorithm for dynamic environments. In *Proceedings of the IASTED Communications and Computer Networks (CCN)*, vol. 1. Citeseer.

Thiagarajan, A., Biagioni, J., Gerlich, T., and Eriksson, J. (2010). Cooperative transit tracking using smart-phones. In *Proc. 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*, p. 85–98, New York, NY, USA.

Wang, Q. and Agrawal, D. Ucbt- bluetooth extension for ns2 at the university of cincinnati. `http://www.ececs.uc.edu/~cdmc/ucbt/`. Accessed: 2014-03-30.

Wang, Z. W. Z., Thomas, R., and Haas, Z. (2002). Bluenet - a new scatternet formation scheme. *Proc. of the 35th Annual Hawaii Int. Conf. on System Sciences*, 00(c):1–9.

Waze Mobile. Waze. `https://www.waze.com/`. Acessado em: 2012-12-12.

Zaguia, N., Daadaa, Y., and Stojmenovic, I. (2008). Simplified bluetooth scatternet formation using maximal independent sets. *Proceedings - CISIS 2008: 2nd International Conference on Complex, Intelligent and Software Intensive Systems*, pp. 443–448.

A Bayesian Game Based Optimization Strategy Proposal for Routing in Energy Constrained DTNs

Sergio Luiz F. Maia[1], Éderson R. Silva[2], Paulo R. Guardieiro[2]

[1]Campus Uberlândia – Instituto Federal do Triângulo Mineiro
Uberlândia – MG – Brazil

[2]Faculdade de Engenharia Elétrica – Universidade Federal de Uberlândia
Uberlândia – MG - Brazil

sergio@iftm.edu.br, ersilva@eletrica.ufu.br, prguardieiro@ufu.br

Abstract. *In this paper, we propose an optimization strategy to be applied to a well-known DTN routing algorithm as PRoPHET and SimBetTS which, by default, don't regard to the issue of energy constraint. Our proposed strategy is based on modeling of the message forwarding as a Bayesian game that aims specifically to capture the dynamic nature of the multi-copy replication decisions, the energy constraint of the nodes and the belief about the energy of other nodes. In addition, we consider factors of evidence aging on accumulated observations used to update the belief that a node has about the energy of the other nodes. The main feature of this belief update system is not to utilize neighborhood watch or acknowledgment mechanism. Moreover, in this paper, we conduct simulation experiments to evaluate the performance of our optimization strategy proposal from a DTN scenario with heterogeneous nodes based on realistic human mobility traces. Simulations results show that our proposed optimization strategy is able to lead the network to remain operational for a longer period of time and, consequently, to achieve a higher final delivery ratio even when compared to a proposal using energy-aware routing.*

1. Introduction

Currently, the research community in communication networks has given special attention to the study of emerging wireless networks such as delay and disruption tolerant networks (DTNs). These networks do not require the presence of a communication infrastructure and often present limited connectivity and the interruptions are frequent. Most DTN routing algorithms employ the technique of store-carry-forward and can use multiple copies of the same message to increase the probability of at least one being delivered. The typical state-of-the art routing algorithms combine heuristics and social network structure to decide for forwarding message copies or 'replicas' among the candidate relays according to some utility function. However, a utility-based replication often leads the routing to direct most of the traffic through a small subset of good relays. This unfair load distribution can quickly deplete the constrained resources utilized by mobile devices, e.g., battery.

In DTN routing literature, the problem of optimal forwarding for message delivery in an energy constrained environment is investigated by some works as [Khouzani et al. 2012], [Yoon and Haas 2012], [Li et al. 2010] and [Rango and Amelio 2013] that utilize current energy-state information in making forwarding decisions.

However, most existing energy-aware routing are proposals applicable to routing algorithms that do not consider the candidate node selection, so potentially all nodes may receive the message copy. For this reason, we propose a solution to the problem of energy-efficient opportunistic forwarding for a DTN routing algorithm that uses some utility function based on a number of different parameters (e.g., encounter history, mobility, sociability, etc.) to discover the better relay nodes, but that was originally proposed without regard to the issue of energy constraint. For example, this is the case of the well-known and cited DTN routing algorithms PRoPHET and SimBetTS.

Our proposal is a new optimization strategy based on a game-theoretic model, the energy constrained forwarding game, which considers heterogeneous and energy constrained nodes able to learn the optimal multi-copy forwarding over time. The goal is to increase the operational period of the network from a better control of energy consumption by nodes, thereby improving the message delivery ratio. In the game, nodes are led to potential selfishness as they may refuse to take on the forwarding tokens of a message received to save energy. A forwarding token implies that the node that owns a message can spawn and forward an additional copy of the given message. In [Maia et al. 2014], we preliminary presented our proposed optimization strategy and simulation-based evaluation of the optimization strategy using synthetic simulation based on Community-based Mobility Model (CBM). However, in order to gain better understanding of our proposal, we extend the considerations about the assumptions, formalizations and definitions of the game-theoretic model.

Additionally, we improve the system to update the belief that a node has about the energy of the other nodes. Due to the dynamic governing the energy consumption, we propose factors of evidence aging on the accumulated observations which are used by the belief update system. Moreover, we present simulation-based evaluation using trace-driven simulation. The used trace captures some human-mobility environments from the Santa Mônica campus of the Federal University of Uberlândia, Brazil. In the end, we present the results of simulation comparing CBM model and mobility trace.

The remainder of this paper is organized as follows. The next section describes related work. In Section 3, we present the formal aspects of the energy constrained forwarding game and the behavior strategies for the nodes are discussed in Section 4. The simulation setup is presented in Section 5 and analysis of the results in Section 6. Section 7 concludes with a discussion of the paper.

2. Related Work

Since the decision makers in a wireless network are devices that need to deal with limited resources, this imposes a conflict of interest. As a result, in order to conserve their resources, the devices may decide not to participate in the process of forwarding the messages, i.e., they adopt a selfish behavior. The theoretical analysis of game-based incentive schemes to stimulate and reward cooperation for routing messages between DTN nodes has been a very active area of research (see, e.g., [Chen and Chan 2010] and [Ning et al. 2011]). In some cases, the analyses include detection of malicious and selfish behaviors of nodes and the implementation of punishment mechanisms [Altman 2009] and reputation spreading of nodes [Gao et al. 2012]. Different from existing incentive schemes which rely on neighbors' monitoring, scoring targets or rewards, our proposed optimization strategy is based on a dynamic non-cooperative game that uses

the history of selected actions and the threats from others nodes in order to encourage the message forwarding. Therefore, we admit that the nodes may become aware of at least part of past behavior of the nodes' forwarding competency and change their strategies accordingly.

An adaptive learning framework that allows nodes to learn the optimal strategies over time can be seen in [El-Azouzi et al. 2013], where the authors apply the evolutionary game theory to non-cooperative forwarding control of relay nodes in DTNs. The paper presents a general framework for competitive forwarding in DTNs under two hops routing, without considering the candidate node selection. In opposition, our model presents forwarding in DTN under multi-hop and replications based on a utility function $U(\cdot)$.

The repeated traditional Bayesian game-theoretic model with an adaptive learning process reported in [Nurmi 2006] for an ad hoc network composed of selfish nodes is the closest one to our game model. However, we use an appropriate repeated *Bayesian signaling game* formulation for copies forwarding, in a heterogeneous and energy constrained DTN. Differently from ad hoc network, in DTN, due to long propagation delay, frequent disconnection, and opportunistic or predicable connections, the assumption that there is a contemporaneous end-to-end path may not be true. Therefore, the mechanisms to the update of the beliefs used by [Nurmi 2006] as watchdog mechanism and acknowledgement messages (ACKs) are not easy to be employed in DTNs. Such mechanisms are not utilized in our proposed belief update system presented in Section 3.

3. Model of the Energy Constrained Forwarding Game

In this section, we present assumptions based on which our forwarding game in energy constrained DTN was developed. Moreover, the interactions between the nodes in a communication opportunity are formalized and the game model is defined. The scenario consisting of quite heterogeneous and sparse node populations, where a limited budget of L message copies needs to be distributed to L relays. Then, the routing algorithm uses some optimization criterion to distinguish the "better" relays and avoid using the least useful ones. Table 1 lists the main variables and simulation parameters commonly used in this paper.

3.1 Message Generation and The Node's Willingness

In this paper, it is assumed the message to be transferred needs to be split into K smaller units called chunks as in [Altman 2009]. The chunks are forwarded independently of the others and can use different intermediate nodes. Only once a sequence of chunks corresponding to a message is received, the message is considered received by the destination node. This approach is used when the contacts between nodes have a finite duration, the file is large with respect to the buffering capabilities of nodes or when a subset of chunks are necessary to reconstruct the message due to redundant information inserted into the coding of the chunks generated. However, the assumption adopted in our proposal is that the number of successes and failures of delivery of a sequence of chunks is used by the destination node to estimate the willingness of intermediate nodes to participate in a multi-hop forwarding process.

Table 1. List of commonly used variables and simulation parameters

Variable/ Parameter	Description	Used Main Value
$U_i(d)$ $[U_j(d)]$	Reflects the fitness or utility that node i $[j]$ will be able to make a delivery to destination node d	
θ	Energy class of a node	Ten classes: {0.1, 0.2, 0.3, …, 1}
$p(\theta)$	Prior probability	Beta(45,15)
K	Total chunks of each message	10
L	Total forwarding tokens for each chunk	16
	Total generated messages	600
s and f	Successes and failures of delivery of K chunks	
$\hat{\theta}_j$	Belief of node i about the energy class of node j	
$\psi(\theta)$	Probability to accept messages according to θ	$exp(-(1.0-\theta)/1.15)$
π	Behavior strategy	
α and β	Parameters of a beta distribution	
$l_i(c)$ $[l_j(c)]$	Forwarding tokens sent [accepted] by node i $[j]$	
ω	Discount weight	0.98
c_1	Parameter of the payoff function	1.4

3.2 Bayesian Signaling Game

Our proposed game considers that the general model is a Bayesian repeated game where each repetition is a stage. The stages of the game begin as simple Bayesian signaling games [Fudenberg and Tirole 1991] played by two nodes when they encounter and a node has chunk copies that can be relayed by other. Bayesian games (also known as games with incomplete information) are models of interactive decision situations in which decision makers (players) have only partial information about the data of the game and about the other players. Adopting a Bayesian statistical approach, we assume that the player who has partial knowledge about the game data has some beliefs or uncertainties about unknown parameters. Players choose their actions during the game according to their beliefs and private information.

One of the most common applications of incomplete information games is signaling games. In a typical signaling game there are two players, one called sender and the other called receiver. The sender observes its type θ from a set of types $\boldsymbol{\Theta}$ and sends a signal m_{sig} to the receiver selected from a set of signals $\boldsymbol{S}_\theta$. The receiver player observes the signal m_{sig}, but not the type θ of the sender, and decides to choose an action a from its action set $\boldsymbol{A}$.

3.3 Game Specification

Let node i be a node that has chunk copies to be forwarded to another node called j. Considering that the nodes will make decisions as in a signaling game, the player node j is the sender (of the signal) and player node i is the receiver. Regarding the type of player, the model assumes that each node in the DTN has a discrete representation for

energy, which is uniquely known to him. This discretized energy is the energy class θ of a node and identifies the concept of type of the player in Bayesian games. The set Θ consists of all possible values of energy. In the admitted signaling game, node i can not observe the type of player j, however, i has a prior knowledge that j can be of type θ with probability $p(\theta)$. The probability $p(\theta)$ is called a prior probability or initial belief of i about the type of j. The prior probability distribution $p(\cdot)$ over Θ is common knowledge among the players, that is, it is assumed that the discretization is global. This means that all nodes in the network can have the same set of possible values for the energy classes.

In the game, node i 'looks for' the best candidates among the nodes of the network that it can trust to replicate the chunks it stores. A new stage of game begins when node i encounters a node j with no copy of a certain chunk m and mobility-aware or social-aware utility function $U_j(d)$, which reflects the probability of node j to deliver this chunk to node d. Thus, node i decides to forward a copy of chunk m to node j according to the criterion of relative utility $U_j(d) > U_i(d)$. Accordingly, this paper assumes that the value $U_j(d)$ for the chunk can be used by node i as a signal of node j's ability to replicate the copies. Then, the set of signals for node j is given by $S_{\theta_j} = \left[U_j^{min}(d), U_j^{max}(d)\right]$, where min and max values for U_j are given by the routing algorithm used. For example, in PRoPHET, U^{min} and U^{max} are 0 and 1, respectively. Node i can use the observable value of the utility function of node j to form a judgment about the real energy class of node j. At an encounter, the updated belief that node i has about the energy class of node j just by observing the value of the utility function is denoted by $\mu\left(\theta_j | U_j(d)\right)$.

The action space of node i derives from the possibilities of number of forwarding tokens for a chunk copy that the node has to grant to node j. Thus, for the chunk m stored in the buffer with $c > 1$ forwarding tokens, the action space of node i is the set $A_m^i = \{l_i(c) | 0 \leq l_i(c) \leq c - 1\}$, i.e., the possible amounts of $l_i(c) \in [0, c-1]$ tokens that node i can assign to the copy of chunk m to be sent to node j during the encounter. On the other hand, let $l_j(c)$ be the number of forwarding tokens effectively accepted by node j for the copy of chunk m with $l_i(c)$ forwarding tokens sent by node i. Thus, the action space of the node j is the set $A_m^j = \{l_j(c) | 0 \leq l_j(c) \leq l_i(c)\}$, i.e., it consists of all alternatives between zero (i.e., don't accept m) and $l_i(c)$. Therefore, the actions of node i clearly restrict the actions of node j within the game tree. As an example of this type of situation, Figure 1 shows a tree of a game with a node j that has two possible types and where node i may simply not send a copy of the chunk, i.e., $l_i(c) = 0$, or forward a copy of the chunk with the possibility of granting to node j from 1 to $l_i(c)$ forwarding tokens.

3.4 Belief Update

In this paper, a node cannot know exactly what energy is available to other nodes, because the scenario admits that there is no incentive for other nodes to reveal that information. Moreover, in our game, energy class is considered as a kind of reputation associated with forwarding competency mapped to match an energy class θ. Hence, it is not trustworthy a node to communicate directly to other nodes its own reputation, since reputation is the opinion of one entity towards another based on past experiences.

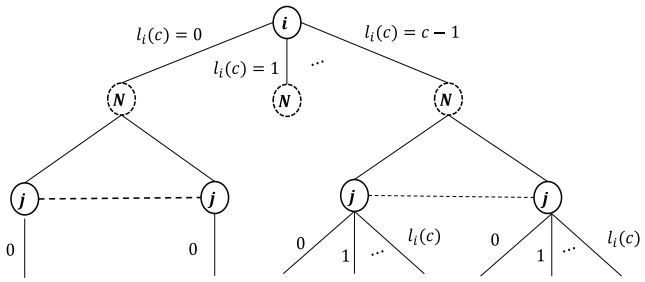


Figure 1. An example of game tree, which has a node *j* with two possible types, and where node i can decide to forward a chunk copy between 0 or $c-1$ forwarding tokens. In a Bayesian game, *N* represents the nature which chooses the type of node *j*, and this type is *j*'s private information.

However, a node can make rough estimates about energy available to other nodes via prior knowledge of the network and own observations gathered during the encounters.

The model considers the chunks of a sequence which were forwarded by an intermediate node as the number of successes and it is denoted by s. Otherwise, the failures f will be given by $K - s$, where K is the number of chunks of a sequence. The chunks delivery of a sequence can occur in a fragmented way through more than one encounter. For this reason, each node that has been the destination of chunk sequences stores in a table the number of successes and failures regarding the nodes participating in the delivery of these sequences. Intuitively, this simply means that the greater the delivery successes assigned to a node, the more likely it is that the node has a greater forwarding competency. Thus, we assume the accumulation of observations of the successes and failures for a participating node can be used to estimate its energy class θ based on Bayesian inference [Neapolitan 2003]. To this end, we can assume that the parameter θ is a random variable and that its prior probability $p(\theta)$ is given by a beta distribution $Beta(\alpha, \beta)$, where α and β are non-negative shape parameters that are initialized to some values α_0 and β_0. When a new observation of s successes and f failures is collected at a new encounter, the prior distribution is updated by $\alpha \leftarrow \alpha + s$ and $\beta \leftarrow \beta + f$.

According to the definition of Bayesian inference, the basic value for a belief b of a reputation can be quantified by the expected value of the beta distribution given by $\alpha/(\alpha + \beta)$. However, this ratio cannot reflect the uncertainty of the distribution. Therefore, the model assumes that node i's opinion about node j is given by $b + \tilde{b} + u = 1$, where b, $\tilde{b}$ and u denote belief, disbelief and uncertainty, respectively. From the Dempster-Shafer theory, the normalized variance of the beta distribution is used to define u as

$$u = \frac{12 \cdot \alpha \cdot \beta}{(\alpha + \beta)^2 \cdot (\alpha + \beta + 1)}. \quad (1)$$

This uncertainty satisfies two important attributes expected for the concept of uncertainty. First, when there is more evidence, u will be consequently lower. Second, when there is greater evidence for s successes or f failures, u will be smaller than when compared to the situation in which both evidences are equal. The total certainty is $(1 - u)$, which is used as the proportion of supporting evidence in the observed results

given by belief. Thus, according to the Bayesian inference, we have that the belief is given by

$$b = \frac{\alpha}{(\alpha + \beta)} \cdot (1 - u). \tag{2}$$

In this paper, in addition to adding uncertainty to the opinion of the node on the estimation of the energy class of other nodes, we include two factors of evidence aging which didn't mention in our previous paper [Maia et al. 2014]. Intuitively, it is expected that the energy value fades along the time because the nodes are continually expending energy on message transmissions. This fading should be considered even that the nodes do not receive messages for updating the prior distribution or that, by the nature of DTNs, the observations are made between long intervals. Thus, the model considers a discount and addition weight, i.e., $\mathcal{D}$ and $\mathcal{A}$, to indicate the freshness, respectively, of success and failure evidences as aging factors for a window of time ΔT, such that

$$\alpha = \alpha \cdot \mathcal{D} + s \quad \text{and} \quad \beta = \beta \cdot (1 + \mathcal{A}) + f, \tag{3}$$

where $\mathcal{D} = \omega^{\frac{\Delta t}{\Delta T}}$, $\mathcal{A} = 1 - \mathcal{D}$ and Δt is the time interval since the last observation, since ω is a discount weight defined empirically and ΔT is the average time interval from an empirically defined number of time intervals between updates of evidence. Therefore, the factors of evidence aging considers that over time the success evidences tend to decrease and the failure evidences, to increase.

4. Strategies of the Proposed Game

In a signaling game, it is possible to classify the equilibria in pure strategy in two extreme cases. The equilibrium is called a separating equilibrium if each type of sender player behaves differently, sending different signals. If the types behave in the same way in the equilibrium, i.e., the same simple signal is sent by all types with probability equal to 1, the equilibrium is called pooling equilibrium. In this section, the behavior of nodes i and j are described considering that the model assumes that the nodes decide on strategies in a pooling equilibrium.

Behavior of node j – The model assumes that at each stage in the first period of game node j observes its type θ_j and sends the signal $U_j(d)$. Then node i observes $U_j(d)$ and acts deciding on the value of $l_i(c)$ forwarding tokens for a chunk copy to be forwarded to node j. In a second period of the game, node j decides how many of the $l_i(c)$ forwarding tokens that have been granted it must accept. For each period, node j has its behavior strategies given by the set $\boldsymbol{\pi}_j = \{\pi_j^1(\cdot), \pi_j^2(\cdot)\}$, where $\pi_j^1(\cdot)$ is the strategy for the first period and $\pi_j^2(\cdot)$ is the strategy for the second period.

So, let $\pi_j^1\big(U_j(d)|\theta_j\big)$ be the probability with which node j of type θ_j sends the signal $U_j(d)$. The model assumes that, in the first period of each stage, the signal $U_j(d)$ can be sent by any type θ_j of node j, i.e. $\pi_j^1\big(U_j(d)|\theta_j\big) = 1$ (pooling equilibrium). Therefore, node i understands that the received signal may have come from a node of any energy class because we assume that the utility function is no energy-aware.

It is assumed that the behavior strategy of node j in response to the action of i, i.e., $\pi_j^2(\cdot)$, is due to a probabilistic event. The event is given by the probability of node j

to accept the forwarding tokens granted by node i to it for the chunk copy sent. The probability of accepting the forwarding tokens is due to the energy class θ_j at the moment of the encounter. Energy management is fundamentally concerned with energy spending on future transmissions of copies that are associated with forwarding tokens of the copy received. So, let $\psi(\theta_j)$ be the function that assigns a probability to the current energy class θ_j. It is assumed that the function $\psi(\cdot)$ maintains orderly energy classes, i.e., if θ_{j1} represents less energy than θ_{j2} then $\psi(\theta_{j1}) < \psi(\theta_{j2})$.

Of course, this means that the nodes with more energy will probably have a greater willingness to accept the forwarding tokens associated to copies to be sent. Thus, given the action of node i to grant $l_i(c)$ tokens for each chunk copy, node j will take on average $\psi(\theta_j) \cdot \sum_n l_i(c_n)$ forwarding tokens for all the n copies of forwarded chunks at the time of the encounter. This corresponds to the behavior strategy for node j in the second period of the stage.

Behavior of node i – The model assumes that the actions of node i must balance the dispersion of copies of chunks it carries and the use of energy. For this purpose, node i forms beliefs that guide its decisions about how many forwarding tokens, for each chunk copy, it must grant to node j. These beliefs or theories are about the types of other players and their behavior strategies. Let $\boldsymbol{\phi_j}$ be the theory of node i about the parameters of interest to a node j, i.e.,

$$\boldsymbol{\phi}_j = \{\hat{\pi}_j, \hat{\theta}_j\}, \tag{4}$$

where $\hat{\pi}$ is the belief of node i about the behavior strategies of node j and $\hat{\theta}_j$ is the belief of node i about the enegy class of node j, i.e., the type of node j.

Once the pooling equilibrium is admitted in which all possible types of node j can send the same type of signal $U_j(d)$, a perfect Bayesian equilibrium with this behavior strategy of node j is only consistent with a belief of node i such that, according to Bayes' rule, $\mu\left(\theta_j | U_j(d)\right) = p(\theta_j)$. This means that the node cannot form any other expectation about node j besides the prior distribution that exists between the node types, so the signal is inefficient in revealing the type. For this, we assume that each node update the prior knowledge about other nodes based on the use of Bayesian inference about accumulated observations of successes and failures in the delivery of a sequence of chunks as is shown in Section 3. Hence, according to Bayes' rule, the belief $\hat{\theta}_j$ about the type of node j should be calculated as $\hat{\theta}_j = [\alpha/(\alpha + \beta)] \cdot (1 - u)$.

The model then assumes that node i assigns a probability $\psi(\hat{\theta}_j)$ for each node j, where $\psi(\cdot)$ is a function that assigns a probability to an energy class. This probability is the node i's belief about the node j's capacity to receive and retain a chunk copy with a certain number of forwarding tokens associated to this chunk. In other words, $\psi(\hat{\theta}_j)$ is the node i's belief regarding the behavior strategy of node j in the second period of the game stage. It is assumed that this probability offers a degree of confidence about the value $U_j(d)$ in the sense that corrects it to a new value given by

$$U_j'(d) = U_j(d) \cdot \psi(\hat{\theta}_j). \tag{5}$$

Note that, given a lower probability assigned to node j, the delivery predictability of node for that destination will be corrected downward. It must be remembered that the calculation of the utility function given by the routing algorithm originally proposed does not take into account any energy constraint.

Regarding its own utility function, it is assumed that node i performs a balance on its utility function based on its current energy class θ_i, such that low energy levels for node i compromise its own delivery predictability. Hence, the new value for $U_i(d)$ is given by

$$U_i'(d) = U_i(d) \cdot \psi(\theta_i). \tag{6}$$

After reevaluating the value of its own utility function and the value of the utility function of node j, node i must decide how many forwarding tokens for the chunk copy to be sent it should grant to node j. The model assumes that the actions of node i must balance the dispersion of copies of chunks it carries and the use of energy. For this purpose, we assume that a node i of energy class θ_i and $U_i'(d)$ decides how many tokens $l_i(c)$ for a chunk copy it should grant to a node j maximizing the payoff composed of two components (gain and cost) given by

$$\mathcal{U}_{\theta_i, U_i'(d)}\left(l_i(c), U_j'(d), \hat{\theta}_j\right) = \tag{7}$$

$$\left(U_j'(d)/\left(U_i'(d) + U_j'(d)\right)\right) \cdot l_i(c) - \left(1 - \psi\left(\hat{\theta}_j\right)\right) \cdot exp\left(c_1 \cdot l_i(c)\right),$$

where c_1 is a parameter whose value is set empirically so that the payoff function meets the requirement of being of Neumann - Morgenstern type and reflects the node i's sensitivity to make the decision for an action $l_i(c)$.

The gain component of the payoff equation considers that node i has more to gain by granting a greater number of forwarding tokens when the ratio of the corrected values of utility given by $U_j'(d)/\left(U_i'(d) + U_j'(d)\right)$ is greater. Evidently, for any fixed $U_j'(d)$, when the node i's energy is more low, $U_i'(d)$ will be consequently lower, then such ratio will present greater values and node i will gain if it grants more forwarding tokens. However, with more energy it could experiment to grant fewer tokens and wait for the next encounter opportunities in the hopes that there will be nodes more favorable to the forwarding. On the other hand, for an action $l_i(c)$, it is assumed that if the probability estimation of node j to reject the granted tokens, given by $1 - \psi\left(\hat{\theta}_j\right)$, is greater, also then the cost component is greater. Furthermore, the model guarantees the experiment conditions of the adaptive learning framework [Groes et al. 1999]. According to this theory, node i selects action $\hat{l}_i(c)$ that maximizes (7) with a probability of $1 - \epsilon_k$, where ϵ_k is a sequence of small errors that decreases in relation to the number of contacts. The equilibrium and optimality proofs of the game-theoretic model with the learning process can be seen in [Nurmi 2006].

5. Modeling and Simulations

We conduct simulation experiments to evaluate our optimization strategy applied in PRoPHETv2 [Grasic et al. 2011] and SimBetTS [Daly and Haahr 2009]. We have chosen these algorithms because they are quite popular within the DTN research community. PRoPHETv2 is the old PRoPHET updated with a new transitive update

equation and direct encounter update equation; and SimBetTS is based on social analysis of past interactions of a node. For the experiments of this paper, we developed the DTN simulator presented in [Maia et al. 2013] from the OMNeT++ Simulation Environment, which provides the basic machinery and tools to write network simulations. The main simulation parameters can be seen in Table I.

5.1 Simulation Setup

Differently of our previous paper, we have run simulations based on traces to evaluate how our proposed strategy can optimize message delivery in a DTN with heterogeneous and energy constrained nodes; the used routing algorithm relies on mobility-aware or social-aware utility function. The used trace data represent a human social network of the Campus Santa Mônica of Federal University of Uberlândia, Brazil (area of 280,119 m^2). Forty nodes are uniformly and sparsely distributed as the users of four department buildings of the Campus. A node can move inside its own community for subareas as office rooms of the university department building. Once in a while, the node can leave its community. When the nodes leave their communities, they are directed to points of interest (POIs). In this paper, four locations of Campus are defined as POIs: one university restaurant, one library e two cafeterias. In these POIs, the encounters occur especially with nodes from other communities. Once out of its community, the node can choose to continue to visit other POIs or return to its community of origin.

Four groups of nodes are used in our simulations: fixed nodes, community nodes, local nodes and roaming nodes. Community nodes are users which do not usually go to POIs and spend the day working in their own offices and occasionally visit other offices in their department building. Local nodes are users which can occasionally go out for library or cafeterias. Roaming nodes are users which daily lunch at the university restaurant, take morning and afternoon coffee break at the Campus' cafeterias and also can go to library. Mobile nodes move with maximum speed of 3.0 m/s on paths defined in the form of Campus' map and choose the shortest path based on movement model of Simulation of Urban Mobility (SUMO) [Krajzewicz et al. 2012]. Values used for movement speeds and wait times represent values commonly observed for real users of Campus. A small transmission range of 5 m was adopted so that can be expected the network will be highly partitioned and not clustered. In addition, the simulator considers that when two nodes move within their transmission range they can exchange information successfully.

In the experiments, after a warm-up period of 5 hours, ten messages are sent every minute for 60 minutes. The pair source and destination nodes are chosen randomly and belong to different communities. With this, the most requested nodes to forward chunks between communities are the roaming nodes and these chunk exchanges occur at POIs or on the paths of Campus. Nodes of a particular department building do not roam around other department buildings. Therefore, the deliveries usually do not occur directly from the source node to the destination node, i.e., the chunks are largely delivered in more than one hop. Hence, the generated network traffic in our scenario requires network routing in order to be delivered, which favors the emergence of some relay nodes most requested than others.

In this paper, a node in the network has the prior probability distribution $Beta(45,15)$ of the energy level of all other nodes. The charge for a battery is defined as being able to perform 2400 transmissions. The proposed evaluation is performed

assuming that the energy reserves of the nodes are not replenished. Thus, when a node runs out of energy, it dies and does not forward any chunks nor does it generate new traffic to the network. Although such assumption is not always true for real users of a Campus since they can recharge their batteries, under this assumption, energy-efficient forwarding algorithms can be better evaluated in terms of the balance of energy consumption of nodes and period of time that the network will remain operational. The traces from Campus environment were used because they insert an adequate repeatability and predictability that could leverage a routing algorithm based on mobility-aware or social-aware utility function.

Under the same energy constrained scenario, we compare the performance of PRoPHETv2 and SimBetTS with regard to three different settings: default mode, energy-aware mode and our optimization strategy. The former mode takes routing decisions based on original utility function. The second introduces energy awareness in the routing decision. In this mode, there is a resultant utility function that is the sum of the original utility function defined by routing algorithm and an energy-aware utility function as used in [Chilipirea et al. 2013]. In both modes, the amount of copies is split between two nodes in proportion to their utility function. In this paper, the results are average from five simulation runs, and the error bars in the graphs represent the 95% confidence intervals.

6. Simulation Results Analysis

Initially, we only consider the performance comparisons of the routing algorithms in relation to the network as a whole. For this, the performance metrics observed are average values of the delivery ratio, the delivery delay, the total transmission and the offline nodes ratio at the end of simulated time of 50h, as are shown in Figure 2.

As can be seen from Figure 2a, the use of our proposed strategy reduces the amount of offline nodes by more than 30% when compared to default and energy-aware mode. As are mainly roaming nodes those nodes more saved by our strategy, then the network remains operational for a longer period of time so the shutdown is delayed. This allows more encounters can happen, thus increasing the likelihood of new successful forwarding. Therefore, the gain in deliveries observed in Figure 2b for our proposed strategy, i.e., 20% for SimBetTS and 13% for PRoPHETv2 when compared to default mode, is primarily due to the lower ratio of offline nodes. This, in turn, is result of the optimization of routing algorithms that achieves as minimal energy consumption as possible, while maintaining adaptability to challenges of the DTN scenario considered. Due to its energy-efficient forwarding, our proposed strategy allows the whole network to achieve similar amount of total transmission as in default and energy-aware mode (see Figure 2c), however, without the same ratios of offloads battery observed for these two modes of operation of the routing algorithms.

It is interesting to note that the improvement in average delivery ratio obtained by our strategy results in increased average delivery delay, as is shown in Figure 2d. This result is in agreement with the most of the research on routing algorithms for DTN, which focuses on reducing the total energy consumed for forwarding at the expense of increased delivery delay. Moreover, we observe that energy-efficient forwarding due to our proposed strategy is achieved by slightly better balance of transmissions between the node classes. In both routing algorithms, the proposed strategy increases slightly the

participation of community and fixed nodes on total transmission. How can be seen in Figure 3, our proposed strategy reduces the participation in the total transmission especially for roaming nodes in SimBetTS. In this routing algorithm, it is a common observation the best nodes carry out most the final hop by which the message arrives to destination. Therefore, we observe that when our proposed strategy is used, a portion of those final hops is transferred to the community and fixed node classes.

Finally, Table 2 provides a comparison between the simulation results in CBM model and mobility trace, considering the use of optimization strategy proposal and the maximum simulated time for the respective scenarios. In general, our proposed optimization strategy applied to SimBetTS takes better advantage of the higher number of contacts between the nodes and the greater degree of predictability and repeatability of node mobility in trace-driven simulation. This results in higher average delivery ratio and lower average delivery delay when compared to PRoPHETv2's results.

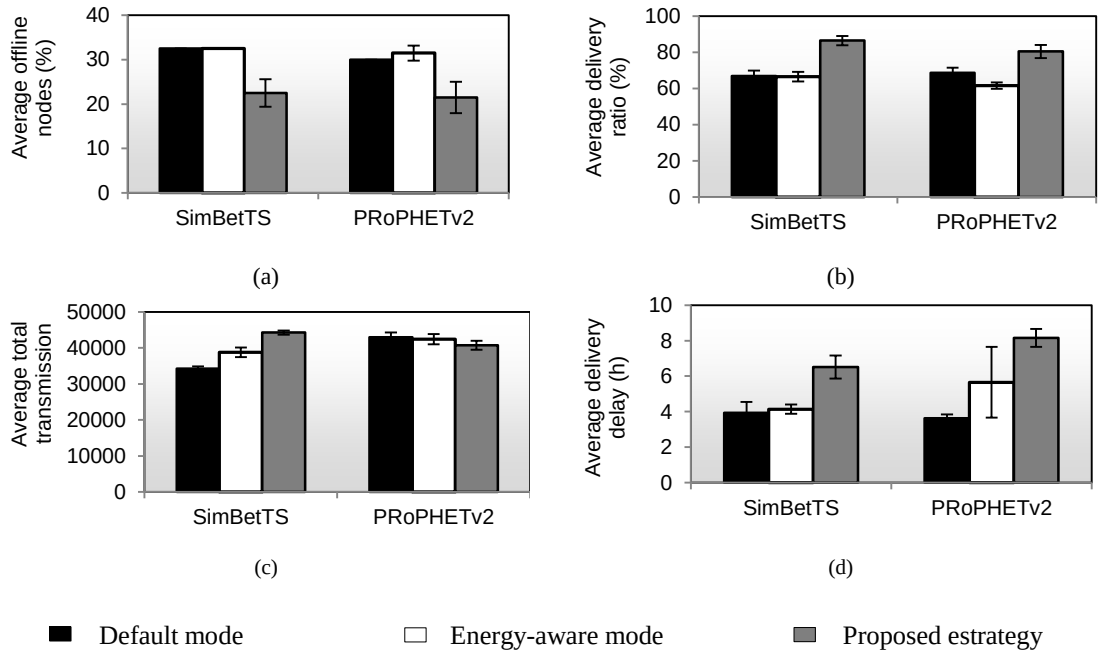


Figure 2. The performance comparisons of the routing algorithms in relation to the network as a whole: a) average offline nodes; b) average delivery ratio; c) average total transmission; and d) average delivery delay.

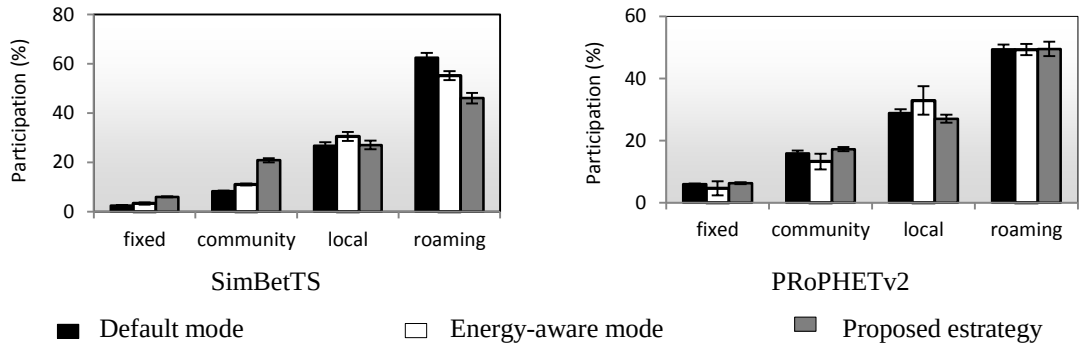


Figure 3. Average participation of the node classes in the total transmission.

Table 2. Result summary of PRoPHETv2 and SimBetTS for CBM and trace at maximum simulated time

Metrics	PRoPHETv2		SimBetTS	
	CBM	Trace	CBM	Trace
Delivery ratio (%)	82	81	83	87
Delivery delay (h)	4	8	3	7
Total contacts	6221	8201	6649	8096
Total transmission	40458	40735	45193	44275

Furthermore, the use of our proposed optimization strategy allows SimBetTS to reduce the total transmission in mobility trace scenario when compared to CBM model.

7. Conclusions

In this paper, we present a new Bayesian game model which optimizes routing in a DTN comprising heterogeneous node populations and subjected to energy constraint. In the proposed game model, a node can learn how to decide on the number of forwarding tokens associated with a chunk to be sent to another node taking into account the estimates created concerning energy and behavior strategies of the other node. The proposed game model can be implemented in DTN scenarios where the routing algorithm uses some utility function to select the best nodes to relay. In addition, we use a system to update beliefs that does not include acknowledgement confirmation or neighborhood watch mechanisms, which are difficult or impractical to be implemented in DTNs. The results of our proposed optimization strategy show a better balance of energy in the network, and keeping it operational for a longer period of time. Social-aware routing as SimBetTS and mobility model based on trace favor our proposed optimization strategy to achieve better performance. For future work, we plan to evaluate other strategies for belief update to lead to even better estimates of the energy classes of network nodes.

Acknowledgments

The authors thank FAPEMIG (under Grant APQ-02117-12) by financial support.

References

Altman, E. (2009). Competition and cooperation between nodes in delay tolerant networks with two hop routing. In *Proceedings of the 3rd Euro-NF conference on network control and optimization (NET-COOP' 09)*.

Chen, B. and Chan, M. (2010). Mobicent: a credit-based incentive system for disruption tolerant network. In *Proceedings IEEE INFOCOM, 2010*.

Chilipirea, C., Petre, A.-C. and Dobre, C. (2013). Energy-aware social-based routing in opportunistic networks. *Int. J. of Grid and Utility Computing*, v. 1, n. 3, p. 1–12.

Daly, E. M. and Haahr, M. (may 2009). Social network analysis for information flow in disconnected delay-tolerant MANETs. *IEEE Transactions on Mobile Computing*, v. 8, n. 5, p. 606–621.

El-Azouzi, R., De Pellegrini, F., Sidi, H. B. A. and Kamble, V. (mar 2013). Evolutionary forwarding games in delay tolerant networks: equilibria, mechanism design and stochastic approximation. *Computer Networks*, v. 57, n. 4, p. 1003–1018.

Fudenberg, D. and Tirole, J. (1991). *Game theory*. Cambridge, MA: The MIT Press. p. 603.

Gao, Z., Zhu, H., Du, S., Xiao, C. and Lu, R. (jun 2012). PMDS: A probabilistic misbehavior detection scheme in DTN. In *IEEE ICC 2012 - Wireless Networks Symposium*.

Grasic, S., Davies, E., Lindgren, A. and Doria, A. (2011). The evolution of a DTN routing protocol – PRoPHETv2. In *Proceedings of the 6th ACM Workshop on Challenged Networks (CHANTS '11)*.

Groes, E., Jacobsen, H. J. and Sloth, B. (1999). Adaptive learning in extensive form games. *Economic Theory*, v. 13, n. 13, p. 125–142.

Khouzani, M. H. R., Eshghi, S., Sarkar, S., Venkatesh, S. S. and Shroff, N. B. (2012). Optimal energy-aware epidemic routing in DTNs. In *Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*.

Krajzewicz, D., Erdmann, J., Behrisch, M. and Bieker, L. (2012). Recent Development and Applications of SUMO – Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, v. 5, n. 3, p. 128–138.

Li, Y., Jiang, Y., Jin, D., et al. (2010). Energy-efficient optimal opportunistic forwarding for delay-tolerant networks. *IEEE Transactions on Vehicular Technology*, v. 59, n. 9, p. 4500–4512.

Maia, S. L. de F., Silva, E. R. and Guardieiro, P. R. (2013). A Proposal of a simulator based on OMNeT ++ for delay / disruption tolerant networks comprising population of nodes with high level of heterogeneity. In *Proceedings of V International Workshop on Telecommunications (V IWT)*.

Maia, S. L. F., Silva, É. R. and Guardieiro, P. R. (2014). A new optimization strategy proposal for multi-copy forwarding in energy constrained DTNs. *IEEE Communications Letters*, v. 18, n. 9, p. 1623–1626.

Neapolitan, R. E. (2003). *Learning Bayesian Networks*. Chicago, IL: Prentice Hall. p. 674.

Ning, T., Yang, Z., Xie, X. and Wu, H. (2011). Incentive-aware data dissemination in delay-tolerant mobile networks. In *8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*.

Nurmi, P. (2006). Modeling energy constrained routing in selfish Ad Hoc networks. In *Proc.from the 2006 workshop on Game theory for communications and networks*.

Rango, F. De and Amelio, S. (2013). Performance evaluation of scalable and energy efficient dynamic n-epidemic routing in delay tolerant networks. In *Inter. Symp. on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*.

Yoon, S. K. and Haas, Z. J. (2012). Energy-aware routing protocol for delay tolerant networks. In: Vasilakos, A.; Zhang, Y.; Spyropoulos, T. V[Eds.]. *Delay Tolerant Networks - Protocols and Applications*. New York: CRC Press. p. 69–100.

Heurísticas para Mapeamento de Redes Virtuais de Sincronia Híbrida

Rômulo Reis De Oliveira[1], **Rasha Ghassan Hasan**[1], **Fernando Luís Dotti**[1]

[1]Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Brazil – Av. Ipiranga, 6681 Prédio 32 sala 505

romulo.reis@acad.pucrs.br, rasha.hasan@acad.pucrs.br

fernando.dotti@pucrs.br

Resumo. *Mapeamento de redes virtuais é um processo impactado pelos requisitos da aplicação. Recentemente, Redes Virtuais de Sincronia Híbrida (RVSHs) foram propostas e sua novidade é a habilidade de fornecer subconjuntos de recursos que se comportam de maneira síncrona, assim podendo suportar aplicações distribuídas que possam necessitar de sincronia parcial. Mapeamento de RVSHs já foi abordado pela literatura por meio de um modelo matemático para encontrar a solução ótima, mas essa abordagem demonstrou precisar de um tempo computacional considerável. Este trabalho apresenta e discute quatro heurísticas para o mapeamento de RVSHs: Duas heurísticas obtidas da adaptação dos algoritmos clássicos, worst fit e best fit, e duas heurísticas que consideram a topologia e demandas da conectividade durante o mapeamento. Os resultados semi-ótimos foram comparados a solução ótima e demonstraram considerável ganho de duas heurísticas em comparação as outras duas em termos de custo de mapeamento e taxa de aceitação.*

Abstract. *Virtual networks embedding is a process that is impacted by the application requirements. Recently, Hybrid Synchrony Virtual Networks (HSVNs) were proposed and their novelty is the ability to provide subsets of resources that behave synchronously, and thus may support distributed applications that may need partial synchrony. HSVNs embedding has already been tackled in the literature through a mathematical model for optimal solution, but this approach showed to need considerable computational time. This work presents and discusses four heuristics for HSVNs embedding: two heuristics obtained by the adaptation of the classic algorithms, worst fit and best fit, and two heuristics that consider topology and connectivity demands while mapping. The semi-optimal results are compared to the optimal solution and showed considerable gain of two heuristics in comparison to the other two in terms of mapping cost and acceptance ratio.*

1. Introdução

A virtualização de redes tem sido considerada a provável solução da ossificação da Internet [Anderson et al. 2005], [Chowdhury and Boutaba 2010], [Feamster et al. 2007] e [Turner and Taylor 2005], pois permite a coexistência de múltiplas redes virtuais em um mesmo substrato físico compartilhado, o que torna possível criar ambientes para avaliar

novas arquiteturas e protocolos. Além disso, essa tecnologia pode ser integrada às arquiteturas atuais. Neste sentido, Redes Virtuais são oferecidas a usuários que as utilizam para executar suas aplicações distribuídas.

Entre os diferentes requisitos de aplicações distribuídas à infraestrutura subjacente, a sincronia (quando o limite máximo de tempo de comunicação e processamento é conhecido) é um aspecto importante, pois determina o conjunto de algoritmos que podem ser empregados [Schneider 1993]. Ambientes completamente assíncronos tem menos custo, mas deixam maior a complexidade dos algoritmos distribuídos. Por outro lado, ambientes completamente síncronos tem alto custo, mas possibilitam a simplificação dos algoritmos. Naturalmente o uso parcial de sincronia é considerado na literatura [Cristian and Fetzer 1999], [Veríssimo 2006]. Quando um subconjunto de recursos tem comportamento síncrono, estes podem hospedar partes cruciais do sistema distribuído fazendo uso da sincronia e provendo benefícios às outras partes do sistema que podem estar executando em recursos que se comportam de forma assíncrona. Um detector de falhas perfeito é um exemplo de aplicação que se beneficia desse tipo de arquitetura [de Araujo Macedo and Gorender 2009].

Em [Hasan et al. 2014b] são propostas as RVSH, redes virtuais que oferecem as abstrações adequadas para a demanda de redes com subconjunto de recursos que se comportam de maneira síncrona. O problema do mapeamento de redes virtuais deste tipo é definido e resolvido de forma ótima, sendo explorado tanto em [Hasan et al. 2014b] quanto em [Hasan et al. 2014a]. Entretanto, devido à potencial escala deste tipo de problema, que é NP-Difícil [Andersen 2002], e da necessidade de solução do mapeamento em curto período tempo, algoritmos heurísticos de mapeamento tem de ser considerados.

Neste artigo são apresentadas quatro abordagens heurísticas adaptadas para solucionar o problema de mapeamento de redes virtuais de sincronia híbrida. A partir dos resultados de um conjunto de experimentos, as soluções heurísticas são comparadas e analisadas com a solução ótima e entre si. Nossos resultados mostram que além do tempo de execução necessário para encontrar uma solução válida ser minimizado pela adoção das heurísticas, algumas destas heurísticas podem obter soluções com o mesmo custo de mapeamento que a solução ótima em alguns cenários.

Este artigo está organizado da seguinte maneira: a Seção 2 revisa trabalhos relacionados, enquanto a Seção 3 apresenta quatro heurísticas para o mapeamento de RVSH. A Seção 4 discute os resultados e por fim é concluído o artigo na Seção 5.

2. Trabalhos Relacionados

Mapeamentos de redes virtuais em redes físicas são extensamente estudados e uma visão geral destes pode ser encontrada em [Fischer et al. 2013]. Os mapeamentos variam em sua definição conforme os objetivos das redes e do mapeamento em questão. Por exemplo, [Zhu and Ammar 2006], [Yu et al. 2008] e [Lu and Turner 2006] consideram questões topológicas no mapeamento, já [Yu et al. 2011], [Rahman and Boutaba 2013], [Chen et al. 2010] e [Oliveira et al. 2013] consideram resiliência, [Bays et al. 2012] segurança e [Hasan et al. 2014b, Hasan et al. 2014a] sincronia. Em algumas propostas, também foram assumidas algumas hipóteses sobre o ambiente para minimizar o espaço do problema. [Zhu and Ammar 2006] e [Lu and Turner 2006], por exemplo, tratam as requisições de redes virtuais de maneira *offline*, onde todas as requisições de redes virtuais já são previamente conhecidas.

O uso de heurísticas para o mapeamento de redes virtuais é importante devido à complexidade computacional para calcular a solução ótima. Neste trabalho, considera-se trabalhos diretamente relacionados as propostas de heurísticas que consideram tempo de atraso. Neste contexto específico enquadra-se [Zhang et al. 2010], onde é proposto um algoritmo heurístico para o mapeamento de redes virtuais orientadas a serviços de *multicast* sujeitos a atrasos e variações de atraso. Nessa arquitetura há um recurso emissor e vários recursos receptores. Jogos online e teleconferência são exemplos de aplicações que se beneficiariam desse tipo de rede virtual. Nessas aplicações é esperado que as mensagens sejam entregues dentro de um período máximo de tempo e que a diferença entre os tempos de entrega nos múltiplos receptores seja mínima. [Zhang et al. 2010] apresenta um modelo matemático cujo o objetivo é minimizar o uso de recursos físicos necessários para efetuar o mapeamento. Também é considerado a capacidade e demanda de CPU e largura de banda dos recursos físicos e virtuais, além do atraso entre o emissor e cada receptor, assim como a variação de atraso entre os receptores. Posteriormente [Zhang et al. 2010] propõe uma heurística com base no modelo matemático, tendo como entrada uma requisição de criação de rede virtual com os seguintes parâmetros: recurso emissor, lista dos recursos receptores, capacidade de CPU dos nodos, capacidade de largura de banda dos enlaces, atraso máximo entre o emissor e receptor e a variação de atraso máximo permitido entre os receptores. Primeiramente são calculados *k shortest paths* entre o emissor e cada receptor, depois são filtrados a fim de localizar o melhor resultado.

O conceito de Redes Virtuais de Sincronia Híbrida (RVSH) foi proposto em [Hasan et al. 2014b, Hasan et al. 2014a], onde RVSHs são redes virtuais que têm subconjuntos de nós e enlances que obedecem a limites de tempo para o processamento e comunicação. As RVSHs são voltadas basicamente para os sistemas distribuídos que podem se beneficiar de sincronia híbrida [Cristian and Fetzer 1999, Veríssimo 2006]. Em [Hasan et al. 2014b], os autores propuseram um modelo matemático para o mapeamento ótimo na forma de um programa inteiro misto (MIP) e eles mostraram que o modelo de mapeamento proposto foi capaz de otimizar a utilização de recursos físicos síncronos disponíveis na rede de substrato, visto que os recursos síncronos são mais caros. A principal limitação do modelo proposto era o tempo computacional necessário para se obter a solução ótima, chegando atingir até 24 horas para encontrar uma solução de mapeamento para um experimento considerado pequeno. Neste trabalho, são propostas heurísticas que necessitam de um tempo computacional menor para encontrar uma solução e ao mesmo tempo minimizar o uso dos recursos síncronos.

Este trabalho se difere dos mencionados nos seguintes aspectos: i) O espaço do problema não é minimizado como em [Lu and Turner 2006], pois as requisições são tratadas de maneira *online*, assim como os recursos são considerados limitados, necessitando de controle de admissão; ii) Um subconjunto dos recursos do substrato físico oferece garantia temporal, assim como um subconjunto de recursos das redes virtuais tem como requisito garantia temporal. Diferentemente de [Zhang et al. 2010], onde todos os recursos físicos e virtuais têm garantias ou demandas por garantias temporais; iii) A terceira heurística considera a topologia da rede virtual na etapa do mapeamento dos nodos por meio da construção de uma árvore de alcançabilidade, diferentemente de [Yu et al. 2008] e [Lu and Turner 2006]; iv) A quarta heurística proposta, além de considerar a topologia durante o mapeamento dos nodos virtuais, também considera se os nodos físicos tem enlaces com largura de banda suficiente para mapear os enlaces do nodo virtual.

3. Algoritmos Heurísticos para mapeamento de RVSH

Nesta seção são descritas as notações e definições de variáveis utilizadas nos quatro algoritmos heurísticos para o mapeamento de redes virtuais de sincronia híbrida, também apresentados nesta seção. Sendo esses algoritmos heurísticos: i) centralizados, somente uma entidade executa o processo de mapeamento; ii) estáticos, não há alteração no mapeamento de uma rede virtual após seu mapeamento; e iii) concisos, busca alocar somente os recursos necessários, sem redundâncias [Fischer et al. 2013]. As quatro propostas têm como principais objetivos minimizar o custo de mapeamento dos enlaces e a alocação de recursos síncronos.

Todos algoritmos apresentados efetuam o mapeamento das RVSH conforme elas chegam, seguindo o modelo de fila. Todos também possuem duas etapas, sendo que na primeira etapa é efetuado o mapeamento dos nodos virtuais sobre os nodos físicos. Para um nodo físico ser elegível como hospedeiro, o mesmo deve ter capacidade de CPU disponível suficiente para hospedar um nodo virtual, o qual tem um custo de CPU, além de respeitar os atributos de sincronia. Um nodo virtual síncrono pode ser mapeado somente em nodos físicos síncronos, já os nodos virtuais assíncronos, podem ser mapeados tanto em nodos físicos assíncronos, quanto síncronos. Também não é permitido que mais de um nodo virtual da mesma rede virtual seja mapeado para o mesmo nodo físico, pois isso aumentaria o impacto caso ocorresse um erro ou falha no nodo físico.

Após todos os nodos virtuais serem mapeados, inicia-se a segunda etapa, onde ocorre o mapeamento dos enlaces virtuais sobre os enlaces físicos. Um enlace físico é considerado elegível para hospedar um enlace virtual somente quando o enlace físico tem uma capacidade de largura de banda disponível suficiente para hospedar o enlace virtual, o qual tem um custo de largura de banda. Além de respeitar a sincronia do enlace virtual, pois, os enlaces virtuais síncronos só podem ser mapeados em enlaces físicos síncronos, já enlaces virtuais assíncronos, podem ser mapeados tanto para enlaces físicos síncronos, quanto assíncronos. Como citado anteriormente, um dos objetivos é economizar recursos síncronos (nodos e enlaces), por estes serem mais caros, sendo assim, recursos virtuais assíncronos somente são mapeados em recursos físicos síncronos, caso não haja recursos físicos assíncronos elegíveis disponíveis. Cada enlace virtual pode ser mapeado tanto para um único enlace físico, quanto para um conjunto de enlaces físicos.

3.1. Definição de estruturas de representação

O substrato físico pode ser representado por um grafo não direcional $G_s = (N_s, L_s)$. Este grafo é composto por um conjunto de nodos físicos N_s conectados por meio de um conjunto de enlaces físicos L_s. Cada $n_s \in N_s$ tem uma capacidade limitada de CPU, assim como, cada $l_s \in L_s$ tem uma capacidade limitada de largura de banda.

Uma requisição de criação de redes virtuais de sincronia híbrida, HSVNRequest, é composta por um conjunto de redes virtuais, onde cada rede virtual de sincronia híbrida pode ser representada por um grafo não direcional $G_v^k = (N_v^k, L_v^k)$, onde, N_v^k é um conjunto de nodos virtuais conectados por enlaces virtuais contidos no conjunto L_v^k. O subscrito v indica se tratar de uma rede virtual e o sobrescrito k é o número identificador da rede virtual.

A sincronia dos recursos virtuais pode ser verificada pela função booleana Sync(n_v^k) para os nodos virtuais e Sync(l_v^k) para os enlaces virtuais, ambas funções retornam *true* para recursos síncronos e *false* para recursos assíncronos.

3.2. Heurística HSVN-WorstFit

Em outros artigos, como em [Liu et al. 2011] e [Li et al. 2013] é adotado um algoritmo base para que seja possível efetuar comparações com os algoritmos propostos. Este algoritmo base é composto por duas etapas. Na primeira etapa ocorre o mapeamento dos nodos virtuais utilizando o algoritmo clássico Worst Fit, sendo considerado o custo do nodo virtual e a capacidade de CPU dos nodos físicos. Na segunda etapa é efetuado o mapeamento dos enlaces virtuais utilizando o algoritmo do caminho mais curto de Dijkstra, alocando cada enlace virtual em um caminho composto por enlaces físicos com largura de banda disponível suficiente para hospedar o enlace virtual. Esse algoritmo heurístico foi adaptado para suportar RVSH. Um dos principais objetivos desse algoritmo é efetuar um mapeamento balanceado, por meio de uma distribuição mais igualitária dos nodos virtuais, evitando o sobrecarregamento dos nodos físicos.

No algoritmo base de RVSH, HSVN-WorstFit (*Algorithm*) 1, é dado como parâmetro de entrada um substrato físico com sincronia parcial $G_s = (N_s, L_s)$ e um conjunto de requisições de criação de redes virtuais de sincronia híbrida HSVNRequest. Para cada $G_v^k \in HSVNRequest$, é chamado o procedimento MapNodesWorstFit(N_v^k, G_s), demonstrado no algoritmo MapNodesWorstFit (*Algorithm* 2).

Algorithm 1 HSVN-WorstFit(G$_s$, $HSVNRequest$)

1: **for all** $G_v^k = (N_v^k, L_v^k) \in HSVNRequest$ **do**
2: **if MapNodesWorstFit**(N_v^k, G_s) == $rejected$ **then**
3: **FreePhysicalResources**(G_v^k, G_s);
4: **else**
5: **if MapLinks**(L_v^k, G_s) == $rejected$ **then**
6: **FreePhysicalResources**(G_v^k, G_s);
7: **end for**

Nessa primeira etapa de mapeamento, os nodos virtuais serão alocados de acordo com sua ordem de chegada. O algoritmo MapNodesWorstFit recebe como entrada um conjunto de nodos virtuais de sincronia hibrida e o grafo $G_s = (N_s, L_s)$ que representa o substrato físico. Para cada $n_v^k \in N_v^k$, é verificado a sincronia do mesmo. Caso o nodo virtual seja síncrono, é chamado o procedimento FindSyncWorstFit(n_k^v, G_s), o qual retornará o nodo físico com maior capacidade de CPU disponível, que não tenha hospedado nenhum nodo virtual da mesma rede virtual e que tenha capacidade de CPU suficiente para hospedar n_v^k. Quando for possível localizar um recurso físico elegível, é efetuada sua alocação por meio do procedimento AllocateNode($n_k^v, physicalNode$), o qual é responsável também por subtrair o custo de n_v^k da capacidade de CPU do nodo hospedeiro. Caso não haja um nodo físico que se encaixe nesses requisitos, será retornado a mensagem *"rejected"*.

Quando o nodo virtual for assíncrono, é chamado o procedimento FindAsyncWorstFit(n_k^v, G_s), esse procedimento é semelhante ao FindSyncWorstFit(n_k^v, G_s), porém, nesse caso, será procurado um recurso físico assíncrono. Caso não haja um recurso físico elegível, é chamado o procedimento FindSyncWorstFit(n_k^v, G_s), pois como citado anteriormente, um nodo virtual síncrono pode ser alocado em um nodo físico síncrono. Caso ambos procedimentos não encontrem um candidato elegível, será retornado a mensagem *"rejected"*.

Algorithm 2 MapNodesWorstFit(N_v^k, G_s)

1: **for all** $n_v^k \in N_v^k$ **do**
2: $physicalNode \leftarrow null$;
3: **if Sync**(n_v^k) == $true$ **then**
4: $physicalNode \leftarrow$ **FindSyncWorstFit**(n_v^k, G_s);
5: **else**
6: $physicalNode \leftarrow$ **FindAsyncWorstFit**(n_v^k, G_s);
7: **if** $physicalNode$ == $null$ **then**
8: $physicalNode \leftarrow$ **FindSyncWorstFit**(n_v^k, G_s);
9: **if** $physicalNode$ == $null$ **then**
10: **return** $rejected$;
11: **else**
12: **AllocateNode**($n_v^k, physicalNode$);
13: **end for**
14: **return** $accepted$;

Quando o algoritmo base de RVSH recebe a mensagem *"rejected"* do procedimento MapNodesWorstFit(N_v^k, G_s), significa que não foi possível efetuar o mapeamento de todos os nodos da rede virtual, sendo assim, é chamado o procedimento FreePhysicalResources(G_k^v, G_s). Esse procedimento desaloca os recursos que foram reservados para o mapeamento dessa rede virtual. Entretanto, quando a mensagem recebida for *"accepted"*, será iniciado a segunda etapa do mapeamento pela chamada do procedimento MapLinks(L_v^k, G_s) (*Algorithm* 3).

O algoritmo MapLinks tem como função mapear os enlaces da rede virtual, esse algoritmo tem como estrada o substrato físico, $G_s = (N_s, L_s)$, um conjunto de nodos virtuais com seus respectivos hospedeiros e um conjunto de enlaces virtuais L_v^k. Para cada enlace é verificado se o mesmo é síncrono ou assíncrono. Quando o enlace for síncrono, será chamada a função FindSyncPath(l_v^k, G_s). Essa retorna o menor caminho físico síncrono, que liga os nodos físicos que hospedam os nodos virtuais de origem e destino desse enlace, com largura de banda disponível suficiente para hospedar o enlace virtual.

Quando o enlace for assíncrono, será chamada a função FindPath(l_v^k, G_s), essa função se difere da função FindSyncPath(l_v^k, G_s), pois, encontra o caminho mais curto, podendo ser esse caminho composto por enlaces físicos tanto síncronos quando assíncronos. Entretanto, enlaces síncronos possuem um valor maior, sendo assim, o caminho retornado por essa função só vai conter enlaces físicos síncronos caso não haja um caminho assíncrono, ou o caminho assíncrono seja composto por um número de enlaces assíncrono significativamente maior que de enlaces síncronos, tornando-se mais caro.

Após ser localizado um caminho válido, o mesmo será mapeado, sendo seu custo subtraído dos enlaces físicos hospedeiros através da função AllocatePath($l_v^k, path, G_s$). Caso não haja um caminho válido, l_v^k não será mapeado e será retornado a mensagem *"rejected"*.

Quando o algoritmo base de HSVN recebe a mensagem *"rejected"* do procedimento MapLinks(L_s^k, G_s), significa que não foi possível efetuar o mapeamento de todos os enlaces da rede virtual, sendo assim, é chamado o procedimento

Algorithm 3 MapLinks(L_v^k, G_s)

```
1: for all l_v ∈ L_v^k do
2:     path ← null;
3:     if Sync(l_v^k) == true then
4:         path ← FindSyncPath(l_v^k, G_s);
5:     else
6:         path ← FindPath(l_v^k, G_s);
7:     if path == null then
8:         return rejected;
9:     else
10:        AllocatePath(l_v^k, path, G_s);
11: end for
12: return accepted;
```

FreePhysicalResources(G_k^v, G_s), esse procedimento desaloca os recursos que foram reservados para o mapeamento dessa rede virtual.

3.3. Heurística HSVN-BestFit

O efeito do algoritmo Best Fit é a concentração dos nodos virtuais alocados em uma quantidade menor de nodos físicos, sendo assim, esse algoritmo também foi adaptado para mapear RVSH. Na heurística HSVN-BestFit, foi trocada a chamada da função MapNodesWorstFit(N_v^k, G_s) para MapNodesBestFit(N_v^k, G_s), linha 2 do *Algorithm* 1. Esse procedimento é similar ao algoritmo MapNodesWorstFit (*Algorithm* 2), porém utiliza a função FindSyncBestFit(n_v^k, G_s) para buscar um nodo físico síncrono com a menor capacidade disponível, mas suficiente para efetuar o mapeamento. Também é utilizado a função FindAsyncBestFit(n_v^k, G_s) para buscar um nodo físico assíncrono com a menor capacidade disponível, mas suficiente para efetuar o mapeamento. Já o mapeamento dos enlaces utiliza o algoritmo MapLinks (*Algorithm* 3).

3.4. Heurística HSVN-Virtual Tree Topology (HSVN-VTT)

Ambos algoritmos apresentados anteriormente não consideravam a topologia ou a distância dos nodos virtuais ao efetuar o mapeamento dos mesmos, o que impacta no custo final do mapeamento (ver Seção 4), já que, nodos virtuais vizinhos na rede virtual podem ser mapeados em nodos físicos distantes, obrigando o mapeamento de um único enlace virtual em vários enlaces físicos, consumindo assim mais recursos.

Por esse motivo, foi desenvolvido a heurística HSVN-VTT, a qual considera a distância entre os nodos virtuais durante o mapeamento dos mesmos. Diversos trabalhos na literatura levam em consideração, no momento do mapeamento dos nodos virtuais para nodos físicos, não somente as demandas aos nodos mas também as demandas da topologia da rede virtual e a possibilidade de seu atendimento na topologia física. [Cheng et al. 2011] e [Liu et al. 2011] consideram a quantidade de enlaces, a largura de banda dos vários enlaces e a CPU de cada nodo, durante a escolha de mapeamento de nodos, para aumentar as chances dos nodos virtuais serem mapeados em nodos físicos, cujos enlaces físicos comportem a demanda dos enlaces virtuais do nodo mapeado em nodos físicos mais próximos, baixando o custo do mapeamento dos enlaces virtuais. Já em [Li et al. 2013], os autores, além de trabalharem com estes elementos, consideram

ainda a distância em número de enlaces intermediários entre dois nodos, para a decisão de mapeamento dos mesmos.

[Li et al. 2013] propõe um algoritmo de mapeamento de redes virtuais também dividido em duas etapas onde no início da primeira etapa é construída uma árvore de alcançabilidade com base na rede virtual. Esta árvore é utilizada para efetuar o mapeamento dos nodos. Esta mesma estratégia foi transposta para o caso de HSVN. Sendo assim, essa heurística tem como base uma parte do modelo apresentado em [Li et al. 2013].

Para cada requisição de criação de rede virtual é construída uma árvore de alcançabilidade, onde a raiz será o nodo virtual com a maior demanda de CPU, sendo seus filhos os nodos diretamente ligados à raiz ordenados de maneira decrescente por demanda de CPU. Os demais nodos são mapeados na árvore de maneira recursiva em profundidade, escolhendo o filho com maior demanda de CPU. A árvore estará completa assim que todos os nodos virtuais fizerem parte da dela. A Figura 1 demonstra a árvore obtida a partir de uma requisição de rede virtual.

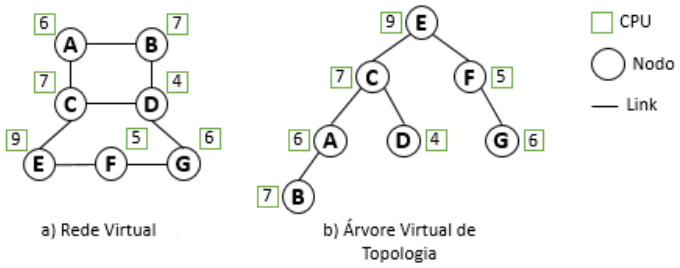


Figura 1. Construção da árvore virtual de topologia a partir de uma rede virtual. Fonte: [Li et al. 2013]

Após a construção da árvore é iniciado o processo de mapeamento dos nodos virtuais, primeiramente é efetuado o mapeando do nodo virtual que está na raiz da árvore. O nodo raiz é mapeado de maneira diferenciada, visto que ele não possui pai. Para evitar gargalos, esse nodo virtual é mapeado para o nodo físico com maior capacidade de CPU disponível e suficiente para hospedar o nodo virtual em questão. O requisito de sincronia também é considerado, sendo que, quando o nodo raiz for síncrono o mesmo só poderá ser mapeado em um nodo físico síncrono, mas se o nodo raiz for assíncrono, ele somente será mapeado em um nodo físico síncrono caso não haja um nodo físico assíncrono elegível.

Posteriormente ao mapeamento do nodo raiz, inicia-se o processo de mapeamento dos filhos em pré-ordem, ou seja, será mapeado os nodos virtuais seguindo um percurso em profundidade com base na árvore criada. Para cada nodo virtual a ser mapeado, é calculado o fator NF para cada nodo físico candidato a hospedar o nodo virtual, lembrando que o nodo pai já tem seu mapeamento decidido. A equação do fator de seleção do nodo (Node Selection Factor - NF) não foi alterada com relação a [Li et al. 2013], ele é calculado com base na capacidade máxima de CPU do nodo físico candidato, e na distância entre o hospedeiro do nodo virtual pai, já decidido, e o nodo físico candidato em questão, conforme a equação (1).

$$NF = MaxCPU(n_s) * (1/Dis(Host(Parent(n_v^k)), n_s)); \quad (1)$$

Seja $n_s \in N_s$ um nodo físico candidato a hospedar o nodo virtual $n_v^k \in N_v^k$;

$Dis(n_s, n'_s)$ a distância entre o nodo físico n_s e o nodo físico n'_s dada pelo número de enlaces entre n_s e n'_s; $Host(n_v^k)$ o nodo físico que hospeda o nodo virtual n_v^k; $Parent(n_v^k)$ o nodo virtual pai de n_v^k na árvore, a equação (1) resulta em um fator NF diretamente proporcional à capacidade de CPU do nodo físico candidato e inversamente proporcional à distância do nodo físico candidato ao nodo físico que hospeda o pai do nodo virtual em mapeamento.

O nodo virtual é mapeado no nodo físico com o maior NF, mas que tenha capacidade de CPU disponível suficiente para hospedar o nodo virtual. Assim, como no mapeamento do nodo raiz, a sincronia também é considerada. Nodos virtuais síncronos são mapeados somente em nodos síncronos, já os nodos virtuais assíncronos são mapeados em nodos síncronos somente se não houverem nodos físicos elegíveis.

A segunda etapa do mapeamento ocorre somente quando todos os nodos virtuais são mapeados com sucesso, nessa etapa inicia-se o mapeamento dos enlaces virtuais, por meio do algoritmo MapLinks (*Algorithm* 3), já descrito anteriormente. Caso não for encontrado um nodo físico ou um caminho elegível, a rede virtual é rejeitada e os recursos antes reservados para a mesma são liberados.

3.5. Heurística HSVN-Virtual Tree Topology & Bandwidth (HSVN-VTTBW)

As heurísticas apresentadas anteriormente neste artigo têm uma deficiência em comum durante a etapa de mapeamento dos nodos. Elas não considerem para cada nodo físico, sua capacidade total de largura de banda e não fazem uma análise de disponibilidade de banda por enlace físico para hospedar um determinado nodo virtual e as demandas de cada um de seus enlaces virtuais. Com isso, uma maior taxa de rejeição acontece na fase de mapeamento de enlaces, já que o mapeamento dos nodos pode ter considerado nodos físicos com enlaces não capazes de suportar as demandas dos respectivos enlaces virtuais.

Com o objetivo de aumentar a taxa de aceitação do algoritmo heurístico anterior, foi incluído neste algoritmo uma nova regra para que um nodo físico seja considerado elegível. Essa regra define que para que um nodo físico seja elegível para mapear um nodo virtual, a largura de banda disponível nos enlaces deste nodo físico devem ser suficientes para que os enlaces do nodo virtual sejam mapeados.

4. Avaliação de Desempenho

Esta seção apresenta uma avaliação das heurísticas propostas, sendo na Subseção 4.1 descrito as ferramentas e parâmetros utilizados nos experimentos e na Subseção 4.2 a apresentação e discussão dos resultados. Os resultados são avaliados com base nos seguintes critérios: 1) taxa de aceitação, 2) utilização dos recursos síncronos, 3) custo do mapeamento dos enlaces, 4) eficiência do mapeamento e 5) tempo de execução.

4.1. Experimentos

Os dados e o ambiente de execução para os experimentos são os mesmos utilizados em [Hasan et al. 2014b], visando uma comparação mais justa dos resultados. Sendo assim, o substrato físico e as redes virtuais foram randomicamente geradas pela ferramenta BRITE[Medina] seguindo o modelo Waxman[Waxman 1988]. Os experimentos foram executados em um computador com CPU de 4 cores de 1.60 GHz, e 2 GB de memória RAM. Foram executados vinte experimentos, divididos em três grupos, A, B e C. Cada grupo com requisições de redes virtuais com tamanho total de 10, 20 e 30 de nodos respectivamente.

Tabela 1. Parâmetros de configuração dos experimentos

Propriedades dos Experimentos				
Grupo : Tamanho da Rede Virtual	A: 10 nodos B: 20 nodos C: 30 nodos			
Cenário	1	2	3	4
Nodos do Substrato Físico	25			
Banda do Substrato Físico	Uniformente distribuído entre 1Gbps e 3Gbps			
CPU do Substrato Físico	100			
Banda das Redes Virtuais	Uniformente distribuído entre 100Mbps e 1Gbps			
CPU das Redes Virtuais	Respectivamente 10, 15 e 25% da capacidade de CPU do substrato físico			
Sincronia do Substrato Físico	30%			100%
Sincronia das Redes Virtuais	0%	30%	60%	x%

Em todos os experimentos, o substrato físico é composto por 25 nodos, sendo a largura de banda dos enlaces uniformemente distribuída entre 1 e 3 Gbps. Nos cenários 1, 2 e 3 o substrato físico é composto por 30% de recursos síncronos, já no cenário 4, 100% dos recursos são síncronos. As redes virtuais são compostas por 3, 4 ou 5 nodos cada, sendo que cada nodo possui respectivamente 10%, 15% e 25% de CPU. A largura de banda das redes virtuais foi uniformemente distribuída entre 100Mbps e 1Gbps. A sincronia das redes virtuais varia de acordo com o grupo, sendo 0% no cenário 1, 30% no cenário 2, 60% no cenário 3, já no cenário 4 a sincronia é referenciada como x%, pois, neste cenário, o custo de mapeamento será independente da sincronia das redes virtuais, visto que o substrato físico será totalmente síncrono.

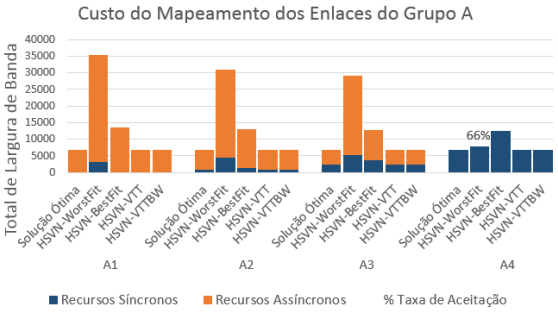
4.2. Resultados

O primeiro fator a ser avaliado é a taxa de aceitação, a qual é obtida pela divisão entre o número de requisições de redes virtuais que puderam ser completamente mapeadas e o número total de requisições. A taxa de aceitação pode ser visualizada nos gráficos de custo de largura de banda dos grupos A, B e C (Figura 2). Para diminuir a poluição visual dos gráficos, a taxa de aceitação só é exibida quando a mesma for inferior à 100%.

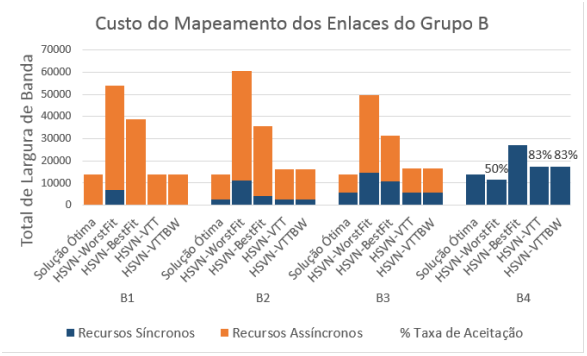
As taxas de aceitação das soluções ótimas foram 100%, demonstrando que é possível efetuar o mapeamento de todas as requisições deste conjunto de experimentos. A heurística HSVN-BestFit apresentou o melhor desempenho nesse critério, obtendo taxa de aceitação inferior a 100% apenas no experimento C4. Já a heurística HSVN-WorstFit teve a pior taxa de aceitação, tendo mapeado com sucesso todas as requisições de apenas um terço dos experimentos. E embora a heurística HSVN-VTTBW tenha obtido uma taxa de aceitação superior à heurística HSVN-VTT no experimento C2, ambas tiveram taxa de aceitação inferior à 100% no grupo C de experimentos.

O segundo fator avaliado foi a utilização de recursos síncronos, pois um dos objetivos das heurísticas propostas é minimizar o consumo desse tipo de recurso. Visto que não há requisição de redes virtuais com demanda de sincronia no cenário 1 e observando os gráficos do custo de largura de banda dos grupos A, B e C (Figura 2), é possível verificar que a as soluções ótimas do cenário 1 não utilizam recursos síncronos, demonstrando que não é necessário a utilização desse tipo de recurso para efetuar o mapeamento. Entretanto, todas as soluções encontradas pela heurística HSVN-WorstFit fazem utilização desnecessária de recursos síncronos, enquanto as demais heurísticas somente utilizam recursos síncronos desnecessários no grupo C, cenário 1, onde as requisições de redes são maiores.

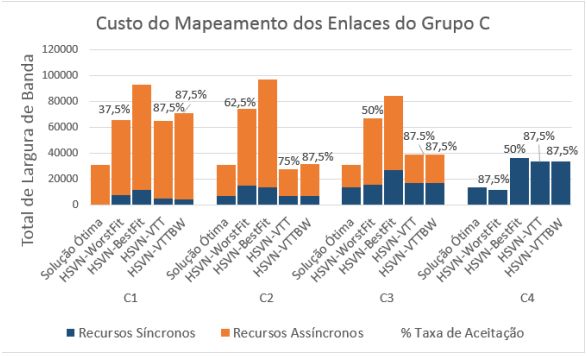
O custo de mapeamento é dado pela soma de CPU e largura de banda utilizada para



(a) Custo de mapeamento do grupo A



(b) Custo de mapeamento do grupo B



(c) Custo de mapeamento do grupo C

Figura 2. Custo de mapeamento dos enlaces de cada cenário por grupo

efetuar o mapeamento. Entretanto, nas heurísticas propostas, o custo de CPU sempre tem o mesmo valor da solução ótima, pois cada nodo virtual pode ser mapeado somente uma única vez e para um único nodo físico. Por esse motivo foi efetuado somente a análise do custo do mapeamento dos enlaces virtuais. Esse custo está diretamente relacionado ao comprimento dos caminhos que conectam dois nodos virtuais.

A heurística HSVN-WorstFit apresentou os custos mais altos, chegando a custar mais de 500% o custo da solução ótima no experimento A1 com 100% de taxa de aceitação. Já as heurísticas HSVN-Tree e HSVN-TreeBW apresentaram os menores custos, inclusive custos mínimos, equivalentes aos custos da solução ótima, em todos os experimentos do grupo A e no experimento B1. Enquanto a heurística HSVN-BestFit apresentou um custo superior em todos os experimentos, porém a melhor taxa de aceitação. Por isso, uma comparação entre as heurísticas com base somente nos custos de mapeamento, sem considerar a taxa de aceitação, pode ser injusto.

Para fazer uma comparação mais justa entre as heurísticas, foi calculado o fator médio de ineficiência de cada algoritmo por grupo (Figura 3). Esse fator é obtido pela divisão entre a soma de largura de banda dos recursos físicos utilizados no mapeamento e a soma da largura de banda mínima necessária para mapear as redes virtuais que foram mapeadas com sucesso. Nos experimentos deste artigo a solução ótima tem fator de ineficiência 1, que é a menor taxa de ineficiência.

O fator médio de ineficiência da heurística HSVN-BestFit aumentou conforme o aumento das redes virtuais, indicando que tal heurística não é a mais adequada para grandes demandas de requisições de redes virtuais. Já as heurísticas HSVN-VTT e HSVN-VTTBW apresentam no grupo A as mesmas taxas de ineficiências que o modelo ótimo. O

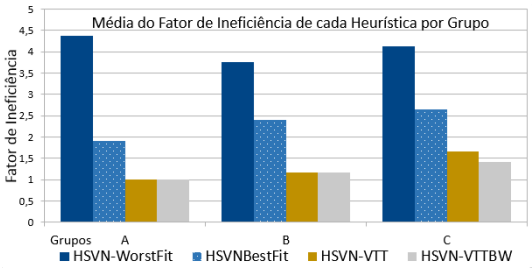


Figura 3. Média de Ineficiência de cada heurística por grupo

que significa que é possível obter-se soluções com custo igual às soluções ótimas com essa heurística. Nos grupos B e C a média de ineficiência apresenta um aumento, mas de qualquer forma ambas as heurísticas obtiveram as menores taxas de ineficiência. No grupo C a heurística HSVN-VTT obteve uma maior ineficiência do que a heurística HSVN-VTTBW, além de uma menor taxa de aceitação, o que indica que que o HSVN-VTTBW tende a ter melhor desempenho para efetuar o mapeamento em redes maiores.

Como pode ser verificado na Tabela 2 do tempo de execução, o modelo de [Hasan et al. 2014b] chegou a levar mais de 58 minutos para encontrar uma solução ótima, enquanto todas as heurísticas aqui propostas encontram uma solução em menos de 1 segundo.

Tabela 2. Tempo necessário para obter a solução ótima de mapeamento, em minutos. Fonte: [Hasan et al. 2014b]

Grupo	Exp.1	Exp.2	Exp.3
A	0.85	0.40	0.31
B	37.55	1.64	10.26
C	58.34	27.12	4.47

5. Conclusão e Trabalhos Futuros

O algoritmo HSVN-WorstFit foi adotado por ser o algoritmo base de outros trabalhos similares, e para o caso de RVSH obteve os piores resultados dentre as outras heurísticas propostas. Já a heurística HSVN-BestFit teve a melhor taxa de aceitação para esse cenário de experimentos, mas o custo da largura de banda e a utilização desnecessária de recursos síncronos cresce significativamente com o aumento das redes virtuais.

As heurísticas de mapeamento HSVN-VTT e HSVN-VTTBW encontraram algumas soluções com o mesmo custo que a solução ótima, tendo o máximo de eficiência para redes virtuais pequenas. Porém apresentam uma redução na taxa de aceitação com o aumento das redes virtuais. O custo de largura de banda e a utilização desnecessária de recursos síncronos demonstrou aumentar conforme o aumento das redes virtuais, entretanto, esse aumento continua sendo inferior ao custo das demais heurísticas. A heurística HSVN-VTTBW obteve taxa de aceitação melhor em um experimento e manteve os mesmos custos nos demais experimentos, indicando que essa heurística alcançou o objetivo de aumentar a taxa de aceitação.

Como trabalho futuro, pretende-se estudar o impacto da topologia das redes físicas e virtuais no desempenho das heurísticas propostas neste trabalho. Esse impacto pode ser avaliado em termos de *(i)* custo de mapeamento, *(ii)* utilização dos recursos síncronos e

assíncronos, e *(iii)* taxa de aceitação. Onde [Luizelli et al. 2013] pode ser considerado um trabalho relacionado.

Agradecimentos

Este trabalho é parcialmente financiado pelo projeto FAPERGS-NPRV (Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul - Núcleo de Pesquisa em Redes Virtuais), projeto PRONEM 11/2038-1. Rasha Hasan é parcialmente financiada por projeto FAPERGS desde 2012 no contexto do seu doutoramento.

Referências

Andersen, D. G. (2002). Theoretical approaches to node assignment.

Anderson, T., Peterson, L., Shenker, S., and Turner, J. (2005). Overcoming the internet impasse through virtualization. *Computer*, 38(4):34–41.

Bays, L., Oliveira, R., Buriol, L., Barcellos, M., and Gaspary, L. (2012). Security-aware optimal resource allocation for virtual network embedding. In *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*.

Chen, Y., Li, J., Wo, T., Hu, C., and Liu, W. (2010). Resilient virtual network service provision in network virtualization environments. In *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*, pages 51–58.

Cheng, X., Su, S., Zhang, Z., Wang, H., Yang, F., Luo, Y., and Wang, J. (2011). Virtual network embedding through topology-aware node ranking. *SIGCOMM Comput. Commun. Rev.*, 41(2):38–47.

Chowdhury, N. M. K. and Boutaba, R. (2010). A survey of network virtualization. *Comput. Netw.*, 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).

de Araujo Macedo, R. and Gorender, S. (2009). Perfect failure detection in the partitioned synchronous distributed system model. In *Proc. of International Conf. on Availability, Reliability and Security (ARES)*, pages 273 –280.

Feamster, N., Gao, L., and Rexford, J. (2007). How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64.

Fischer, A., Botero, J., Till Beck, M., de Meer, H., and Hesselbach, X. (2013). Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906.

Hasan, R., Machado Mendizabal, O., Reis De Oliveira, R., and Dotti, F. (2014a). A study on substrate network synchrony demands to support hybrid synchrony virtual networks. In *Computer Networks and Distributed Systems (SBRC), 2014 Brazilian Symposium on*, pages 344–352.

Hasan, R., Mendizabal, O., and Dotti, F. (2014b). Hybrid synchrony virtual networks: Definition and embedding. In *ICN 2014, The Thirteenth International Conference on Networks*.

Li, X., Guo, C., Wang, H., Li, Z., and Yang, Z. (2013). A constraint optimization based virtual network mapping method. In *2012 International Conference on Graphic and Image Processing*, pages 87683X–87683X. International Society for Optics and Photonics.

Liu, J., Huang, T., Chen, J.-y., and Liu, Y.-j. (2011). A new algorithm based on the proximity principle for the virtual network embedding problem. *Journal of Zhejiang University SCIENCE C*, 12(11):910–918.

Lu, J. and Turner, J. (2006). Efficient Mapping of Virtual Networks onto a Shared Substrate. Technical report, Washington University in St. Louis.

Luizelli, M. C., Bays, L. R., Buriol, L. S., Barcellos, M. P., and Gaspary, L. P. (2013). Characterizing the impact of network substrate topologies on virtual network embedding. In *In Proc. to the 9th CNSM and Workshops (International Conference on Network and Service Management)*, pages 42–50.

Medina, A.; Lakhina, A. M. I. B. J. Brite: Boston university representative internet topology generator.

Oliveira, R., Marcon, D., Bays, L., Neves, M., Buriol, L., Gaspary, L., and Barcellos, M. (2013). No more backups: Toward efficient embedding of survivable virtual networks. In *Communications (ICC), 2013 IEEE International Conference on*, pages 2128–2132.

Rahman, M. and Boutaba, R. (2013). Svne: Survivable virtual network embedding algorithms for network virtualization. *Network and Service Management, IEEE Transactions on*, 10(2):105–118.

Schneider, F. B. (1993). Distributed systems (2nd ed.). chapter What good are models and what models are good? ACM Press/Addison-Wesley Publishing Co.

Turner, J. and Taylor, D. (2005). Diversifying the internet. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 2, pages 6 pp.–760.

Veríssimo, P. E. (2006). Travelling through wormholes: a new look at distributed systems models. *ACM SIGACT News*, 37(1).

Waxman, B. M. (1988). Routing of multipoint connections. *Selected Areas in Communications*, 6(9).

Yu, H., Anand, V., Qiao, C., and Sun, G. (2011). Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In *Communications (ICC), 2011 IEEE International Conference on*.

Yu, M., Yi, Y., Rexford, J., and Chiang, M. (2008). Rethinking virtual network embedding: Substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29.

Zhang, M., Wu, C., Jiang, M., and Yang, Q. (2010). Mapping multicast service-oriented virtual networks with delay and delay variation constraints. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*.

Zhu, Y. and Ammar, M. (2006). Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12.

Tolerância a Faltas Arbitrárias em Processamento Distribuído de Grafos

Daniel Presser[1], **Lau Cheuk Lung**[1], **Miguel Correia**[2]

[1]Departamento de Informatica e Estatística – Universidade Federal de Santa Catarina (UFSC)
88040-900 – Trindade – Florianopolis – Santa Catarina – Brasil

[2]Instituto Superior Tecnico, Technical University of Lisbon, INESC-ID – Lisboa, Portugal

Abstract. *Graphs are used to model a large number of real problems in areas such as machine learning and data mining. The increasing datasets sizes has led to the creation of various distributed large scale graph processing systems, among which Google's Pregel stands out. Although these systems usually tolerate crash faults, literature suggests they are vulnerable to accidental arbitrary faults as well. In this paper we present an algorithm and a prototype of such system that can tolerate this kind of fault, based on GPS, an open source implementation of Pregel. Experimental results of the prototype in Amazon AWS are presented, showing that it uses only twice the resources of the original implementation, instead of 3 or 4 times as usual in Byzantine fault-tolerant systems. This cost is acceptable for critical applications that require this level of fault tolerance.*

Resumo. *Grafos são usados para modelar um grande número de problemas reais em áreas como aprendizado de máquina e mineração de dados. O crescimento das bases de dados destas áreas tem levado à criação de uma variedade de sistemas distribuídos para processamento de grafos muito grandes, dentre os quais se destaca o Pregel da Google. Embora esses sistemas costumem ser tolerantes a faltas de parada, a literatura sugere que eles também estão suscetíveis a faltas arbitrárias acidentais. Neste artigo é apresentado um algoritmo e protótipo de sistema de processamento de grafos distribuído capaz de lidar com essas faltas, baseado no GPS, uma implementação de código aberto do Pregel. São apresentados também resultados experimentais do protótipo obtidos na Amazon AWS, onde demonstra-se que este algoritmo usa apenas o dobro de recursos do original, em vez de 3 ou 4 vezes como é comum em modelos tolerantes a faltas Bizantinas. Com isso, seu custo torna-se aceitável para aplicações críticas que requerem esse nível de tolerância a faltas.*

1. Introdução

Grafos são extensivamente usados na modelagem e solução de problemas reais em áreas como aprendizado de máquina e mineração de dados. Por isso, algoritmos e modelos de processamento em grafos também enfrentam os desafios impostos pelo crescente tamanho das bases de dados e complexidade das análises realizadas por eles. Isto tem chamado a atenção dos pesquisadores para modelos de processamento capazes de lidar com grafos muito grandes de maneira paralela e distribuída. Vários modelos têm sido propostos, como Pregel [Malewicz et al. 2010], PowerGraph [Gonzalez et al. 2012],

Trinity [Shao et al. 2013] e Graphx [Xin et al. 2013]. Dentre estes, destaca-se o Pregel, desenvolvido pela Google. Trata-se de um modelo que pode ser utilizado em *clusters* com milhares de máquinas e é capaz de processar grafos da ordem de bilhões de vértices. A implementação do Pregel é proprietária da Google, mas existem implementações de código aberto, como o Apache Giraph [Avery 2011] e o GPS [Salihoglu and Widom 2013].

Por ser tão escalável, o Pregel tolera faltas através do uso de *checkpoints*. De tempos em tempos, o estado de cada processo é armazenado num sistema de arquivos distribuído, para que fique disponível mesmo que o processo falhe. Caso algum problema seja detectado, os processos restauram seu estado a partir do último *checkpoint* registrado. Se alguma das máquinas não estiver mais disponível, seu *checkpoint* é distribuído para os processos restantes. Após isso, o sistema recomeça o processamento do ponto onde parou.

Tolerar problemas como falha de uma máquina ou de disco é fundamental. Entretanto, há problemas conhecidos e mais sutis, que podem afetar a corretude do resultado de um processamento sem causar sua interrupção [Schroeder and Gibson 2007]. Há vários anos um estudo já mostrava a existência de faltas arbitrárias ou Bizantinas em vários sistemas reais [Driscoll et al. 2003]. Mais recentemente, um estudo conduzido por 2 anos e meio nos servidores da Google evidenciou que mais de 8% das DIMMS apresentaram falhas em algum momento [Schroeder et al. 2009]. Outro estudo da Microsoft mostrou que erros em processadores também são frequentes [Nightingale et al. 2011]. Os mecanismos de tolerância a faltas do Pregel e de suas implementações atuais não são capazes de lidar com falhas arbitrárias acidentais, que podem ser causadas por esse tipo de erro em memórias ou processadores.

Para lidar com esse tipo de falta, normalmente usa-se a replicação do processamento para mascarar seus efeitos. Um exemplo de técnica genérica é o algoritmo de replicação de máquinas de estado conhecido por *Practical Byzantine Fault Tolerance* [Castro and Liskov 2002]. No entanto, por ser voltada para aplicações cliente-servidor, esta técnica é inadequada para processamento distribuído de grafos. Além disso, são necessárias $3f + 1$ réplicas para tolerar f faltas, que torna seu custo proibitivo no contexto de larga escala. Outra técnica mais eficiente é a que foi utilizada para tornar o *MapReduce* [Dean and Ghemawat 2008] tolerante a faltas Bizantinas [Costa et al. 2011]. Neste modelo, que usa $f + 1$ réplicas, cada tarefa *map* ou *reduce* é replicada e seus resultados são comparados. No caso de divergência, a tarefa é executada novamente até que se alcance uma maioria de resultados iguais. Entretanto, um estudo mostrou que o *MapReduce* é até 10 vezes mais lento que modelos como o do Pregel para processamento de grafos [Kajdanowicz et al. 2014], o que inviabiliza seu uso.

Neste artigo é apresentado o *Greft*, um sistema de processamento distribuído de grafos baseado no modelo do Pregel capaz de lidar com faltas arbitrárias acidentais de maneira eficiente. Neste modelo, cada vértice do grafo é replicado em máquinas diferentes para que seja possível comparar a evolução de seu estado durante o processamento. Com o uso de diversas técnicas foi possível reduzir o número de réplicas necessárias para $f + 1$, para tolerar f réplicas faltosas. A verificação da evolução do estado dos vértices é otimizada usando-se resumos criptográficos dos dados das réplicas, reduzindo enormemente o volume de dados trafegados entre os processos. Além disso, foi introduzida uma nova técnica de gerenciamento dos *checkpoints*, que dispensa o uso de um

sistema de arquivos distribuídos nesta tarefa. Valendo-se da replicação e de resumos criptográficos para validar a integridade dos arquivos, os processos podem armazenar os *checkpoints* em disco local sem comprometer as características do sistema. O sistema de arquivos distribuído é usado apenas para armazenar os arquivos de entrada e os resultados da computação.

As principais contribuições deste trabalho são:

1. A criação de um algoritmo tolerante a faltas arbitrárias acidentais para processamento distribuído de grafos;
2. A implementação de um protótipo deste algoritmo com base no GPS, uma implementação do Pregel, incluindo otimizações no gerenciamento dos *checkpoints*;
3. Uma avaliação experimental detalhada do funcionamento e desempenho do protótipo na Amazon AWS, utilizando um grafo real para os testes.

2. Pregel e GPS

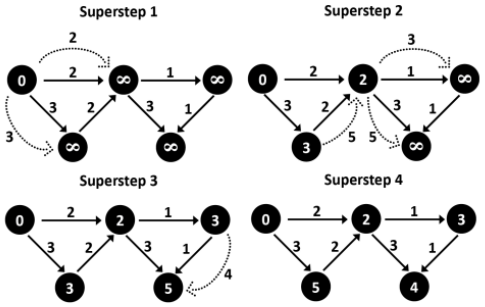
Pregel é um modelo de desenvolvimento e um *framework* que suporta este modelo para processamento distribuído de grafos em larga escala. Neste modelo, cada vértice do grafo possui uma identificação única, um valor, um estado (ativo ou inativo), uma lista de adjacências com ou sem pesos e uma fila de mensagens recebidas. Os vértices são distribuídos entre as máquinas de um *cluster* para realizar o processamento. Cada máquina geralmente processa muitos vértices do grafo.

Os programas são expressos como uma sequência de iterações, chamadas de *supersteps*, nas quais uma função definida pelo usuário é executada em cada vértice do grafo, conceitualmente em paralelo. A função define o comportamento de um único vértice V dentro do *superstep* S. Nela, o vértice V pode ler as mensagens enviadas por outros vértices no *superstep* $S-1$, modificar seu valor e estado, modificar suas arestas e pesos, e enviar mensagens a outros vértices. Estas mensagens só serão recebidas pelo vértice de destino no *superstep* $S+1$. Com isso, garante-se o paralelismo da execução da função nos vértices dentro de um *superstep* mesmo que, na prática, ela ocorra sequencialmente nas máquinas. Este modelo de processamento em iterações é baseado no modelo *Bulk Synchronous Parallel* [Valiant 1990].

Um sistema de coordenação gerencia a execução dos *supersteps* nas máquinas, garantindo que cada *superstep* só inicie depois que todas as máquinas finalizaram o *superstep* anterior. A execução continua até que todos os vértices estejam inativos e nenhuma mensagem esteja na fila de entrega. Este modelo é suficientemente flexível para permitir a implementação de um grande número de algoritmos em grafos. Além disso, o esquema de sincronia facilita a implementação dos algoritmos e garante que o sistema estará livre de *dead-locks* e condições de corrida - comuns em sistemas assíncronos.

A figura 1 ilustra esse modelo com a execução em um grafo do algoritmo *Single Source Shortest Path* [Malewicz et al. 2010], que encontra o menor caminho de um vértice de origem a todos os outros. No primeiro *superstep*, cada vértice inicia seu valor com $+\infty$, exceto pela origem, que se inicia com zero. Ainda nesse *superstep*, o vértice de origem envia mensagens para seus adjacentes com seu valor atual somado ao peso da aresta que os conecta. Estas mensagens serão recebidas pelos destinatários depois que o próximo *superstep* for iniciado, garantindo o paralelismo da execução do algoritmo em

cada vértice durante o *superstep*. No segundo *superstep*, os vértices adjacentes à origem recebem as mensagens e atualizam seu valor com a menor distância entre elas. Os que atualizaram seu valor então enviam mensagens para os seus adjacentes e o processamento continua até que nenhum vértice tenha mensagens pendentes de recebimento.



Single Source Shortest Path

```
1: minDist ← origem(this.id)  0 : +∞
2: for msg em this.mensagens do
3:     minDist ← min(msg.dist, minDist)
4: end for
5: if minDist < this.valor then
6:     this.valor ←minDist
7:     for aresta em this.arestas do
8:         valorMsg ← this.valor + aresta.peso
9:         enviar(aresta.id, valorMsg)
10:    end for
11: end if
12: this.ativo ← FALSE
```

Figure 1. Exemplo de execução do algoritmo num grafo pelo modelo do Pregel

O *Graph Processing System* (GPS) [Salihoglu and Widom 2013] é uma implementação de código aberto de um modelo baseado no Pregel. Ele traz como contribuições melhorias na partição do grafo entre as máquinas e uma interface para computações globais. As contribuições no particionamento tem por objetivo manter vértices que se comunicam muito nas mesmas máquinas, reduzindo a quantidade de mensagens trocadas na rede. Já a interface para computações globais facilita a implementação de algoritmos que seriam muito complicados de implementar com uma visão centrada aos vértices - como alguns algoritmos de clusterização. Sua arquitetura conta com um processo *coordenador*, chamado *GPSMaster* que gerencia a execução dos *supersteps* nos processos, chamados de *GPSWorker*. O *Hadoop Distributed File System* (HDFS) é usado para armazenar e distribuir os arquivos de entrada, os *checkpoints*, e os resultados finais da computação. Nos checkpoints são armazenados todos os dados dos vértices atribuídos ao processo, incluindo a fila de mensagens recebidas para o *superstep* em que o *checkpoint* foi gerado. O HDFS é otimizado para o armazenamento de arquivos de grandes dimensões tolerando faltas de parada [Shvachko et al. 2010].

O processo coordenador é o responsável por detectar eventuais falhas nos outros processos através de troca de mensagens periódicas. Caso um processo falhe, o coordenador solicita aos restantes que reiniciem o processamento a partir do último *checkpoint* armazenado. Os vértices que estavam sob a responsabilidade do processo faltoso são redistribuídos entre os restantes e o processamento é reiniciado.

3. Greft - Processamento de Grafos Tolerante a Faltas Arbitrárias

Nesta seção é apresentado a proposta do Greft, um sistema de processamento de grafos tolerante a faltas arbitrárias acidentais baseado no GPS. Inicialmente são apresentadas a arquitetura e o modelo do sistema, depois os algoritmos e as técnicas usadas.

3.1. Arquitetura

O sistema é composto de um conjunto de processos distribuídos. Os clientes solicitam a execução de processamentos em um grafo e aguardam sua conclusão. O coordenador,

chamado *GPSMaster*, recebe a solicitação do cliente e gerencia a distribuição dos vértices do grafo entre os processos. Nessa distribuição, o *GPSMaster* cria duas ou mais réplicas de cada processo atribuindo a eles os mesmos conjuntos de vértices. Os processos sob supervisão do *GPSMaster*, chamados de *GPSWorkers*, recebem os vértices, realizam a computação de cada *superstep*, e armazenam seu estado nos *checkpoints* quando solicitado pelo *GPSMaster*. Além disso, o sistema de arquivos distribuído utilizado (HDFS) também é composto de dois tipos de processos. O *NameNode* é um processo mestre que gerencia o armazenamento dos arquivos do sistema. Já os *DataNodes* são os processos que armazenam dos dados dos arquivos nas máquinas.

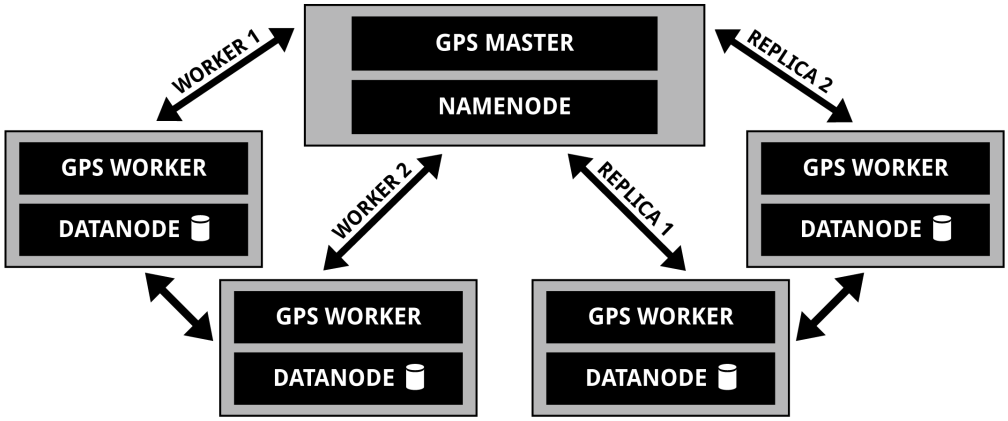


Figure 2. Arquitetura do Greft, baseada no GPS

3.2. Modelo do Sistema

É definido que o processo é correto se ele segue os algoritmos estabelecidos. Caso contrário, trata-se de um processo faltoso. Os processos são executados em servidores em um *datacenter*. O modelo de faltas é o de faltas arbitrárias acidentais, permanentes ou transientes [Avizienis et al. 2004]. Assume-se que um *GPSWorker* pode falhar por parada ou produzir resultados arbitrários, mas não de maneira maliciosa. Já os clientes são tidos como sempre corretos, visto que são os interessados no resultado da computação. O processo coordenador também é assumido como sempre correto, do mesmo modo que no GPS e no Pregel que não possuem mecanismos para tratar falhas no coordenador. Embora tenha sido usada a versão oficial, o HDFS também é assumido como tolerando faltas arbitrárias, pois já existe uma versão do HDFS tolerante a faltas Bizantinas utilizando a biblioteca *UpRight* [Clement et al. 2009].

O sistema é assíncrono, pois não há nenhum pressuposto quanto a limites em tempo de processamento ou comunicação. Há apenas um tempo máximo de espera na checagem de estado feita pelo *GPSMaster* nos *GPSWorkers* para detectar eventuais problemas. Também é assumido que os processos são conectados por canais confiáveis, onde mensagens não são perdidas, duplicadas ou corrompidas - como é provido pelo TCP/IP. Finalmente, também assume-se a existência de funções resumo criptográfico (*hash*), por exemplo o SHA-1, resistentes à colisão, isto é, que seja inviável encontrar duas entradas que resultem na mesma saída.

O algoritmo é parametrizado com f. Normalmente este parâmetro indica a quantidade de processos que podem falhar arbitrariamente sem comprometer o funcionamento

do sistema. Neste caso, o parâmetro tem um significado diferente: assumindo um conjunto $\{V_1, V_2, \cdots, V_n\}$ de réplicas de um vértice V, f é o número máximo de faltas que podem ocorrer nas réplicas de forma que o estado do vértice esteja igual entre elas após a execução de um (ou conjunto de) *supersteps*. O algoritmo também tem os parâmetros f_max, que indica o número máximo de vezes que um conjunto de réplicas pode apresentar divergências antes de ser removido do sistema, e spc, que define o intervalo de *supersteps* entre cada *checkpoint*.

3.3. Algoritmo

Os algoritmos 1 e 2 detalham o funcionamento do *GPSMaster* e GPSWorker. O algoritmo do *GPSMaster* consiste em particionar o grafo de entrada e distribuir cada partição para $f + 1$ *GPSWorkers* diferentes. Cada partição é um subgrafo do grafo original, gerado usando qualquer uma das técnicas de particionamento disponibilizadas pelo GPS. Depois disso, os *supersteps* são iniciados com o envio do comando *START_SUPERSTEP* para os processos. Ao final de cada *superstep*, o coordenador compara o estado de cada vértice com suas réplicas. Caso sejam diferentes, é sinal que o estado de uma das réplicas do vértice foi corrompido durante o processamento e precisa ser recuperado. Para isso, o *GPSMaster* envia o comando *RESTORE_CHECKPOINT* para que os *GPSWorkers* restaurem o último *checkpoint*. Dado que o último *checkpoint* foi feito num momento em que nenhum vértice estava corrompido, ao restaurá-lo todos os vértices voltarão a um estado consistente a partir do qual o processamento pode recomeçar e continuar.

Comparar o estado de cada vértice de um grafo com centenas de milhões de vértices distribuídos em várias máquinas é dispendioso, tanto pelo tempo de comparação quanto pelo volume de dados trafegados até o coordenador. Para evitar isso, cada *GPSWorker* computa um resumo criptográfico dos vértices sob sua responsabilidade ao final de cada *superstep*, e apenas esses resumos são comparados. Isso pode ser visto no algoritmo 2. O que é o estado de cada vértice varia conforme a função definida pelo usuário: pode ser apenas o valor do vértice, ou os pesos das suas arestas, ou ambos.

No caso caso de *faltas transientes* - que acontecem mas depois desaparecem - não há necessidade de remover processos e redistribuir os vértices. Bastaria executar novamente o processamento para tentar chegar a uma maioria de $f + 1$ resultados iguais para cada vértice do grafo. Entretanto, uma *falta permanente* pode fazer com que o conjunto de *GPSWorkers* nunca dê resultados iguais. Para este caso, é definido o parâmetro f_max como o limite a partir do qual o conjunto de réplicas é removido do sistema. Já quando um processo falha por parada (*crash*), ele entra na lista de suspeitos por não enviar as mensagens periódicas de estado (*heartbeats*) ao *GPSMaster*. Neste caso o conjunto de réplicas também é removido do sistema. Quando o GPSMaster remove um conjunto de processos, primeiramente ele verifica se existem máquinas adicionais que possam ser usadas para dar continuidade ao processamento. Se existirem, a partição dos processos removidos é enviada a elas através do comando *PARTITION* e elas são adicionadas ao processamento. Se não houver mais máquinas à disposição, a informação dos *checkpoints* dos processos removidos é incluída no comando *RESTORE_REPLICAS* e enviado aos *GPSWorkers*. Cada conjunto restante, por sua vez, restaura uma partição do arquivo de *checkpoint* do conjunto removido (função *restoreCheckpointPartition* do algoritmo 2), de forma que os vértices presentes no arquivo sejam igualmente distribuídos entre eles e sejam replicados. Em ambos casos o *GPSMaster* também envia o comando *RESTORE_CHECKPOINT* para

que os processos restaurem o último *checkpoint* e recomecem o processamento.

Algorithm 1 GPSMaster (Greft)

```
Require: 𝒲: set of workers, 𝒢: input graph, 𝒩: available nodes
1:  function GPSMaster(f, f_max, spc)
2:      𝒫 ← partition 𝒢 into ⌊|𝒲|/(f + 1)⌋ subgraphs
3:      for all p ∈ 𝒫 do
4:          ℛ[p] ← {w ∈ 𝒲 | w_1...w_{f+1}} //create a set of f + 1 replicas to each partition
5:          𝒲 ← 𝒲 − ℛ[p] //remove used workers from set
6:          ℱ[p] ← 0 //initialize faults per partition
7:      msg ← empty message
8:      msg.addCommand(PARTITION, ℛ) //send partitions to workers
9:      superstep ← 1
10:     while 𝒢.active() do
11:         msg.addCommand(START_SUPERSTEP, superstep)
12:         if superstep mod spc == 0 then
13:             msg.addComand(CREATE_CHECKPOINT, superstep) //create checkpoint if spc interval has passed
14:         sendToAll(ℛ, msg)
15:         msg ← empty message
16:         superstep ← superstep + 1
17:         ∀r ∈ ℛ wait for responses or suspicions
18:         for all p ∈ 𝒫 do
19:             //If there are suspicions of crashes or differences between hashes...
20:             if (∃w ∈ ℛ[p] | suspicions[w] = True) or (∃w, w' ∈ ℛ[p] | resp[w].hash ≠ resp[w'].hash) then
21:                 checkpoint ← lastCheckpoint() //load last checkpoint info
22:                 superstep ← checkpoint.superstep //replace current superstep number
23:                 msg.addCommand(RESTORE_CHECKPOINT, checkpoint) //add command to restore last checkpoint
24:                 ℱ[p] ← ℱ[p] + 1
25:                 if (ℱ[p] > f_max) or (∃w ∈ ℛ[p] | suspicions[w] = True) then
26:                     removed ← ℛ[p]
27:                     ℛ ← ℛ − {removed} //remove replicas from set
28:                     if |𝒩| > f + 1 then
29:                         ℛ[p] ← {w ∈ 𝒩 | w_1...w_{f+1}} //if there are available nodes, use them
30:                         𝒩 ← 𝒩 − ℛ[p] // remove used nodes from set
31:                         ℱ[p] ← 0
32:                         msg.addCommand(PARTITION, ℛ[p]) // Send partition only to new replicas
33:                     else
34:                         msg.addCommand(RESTORE_REPLICAS, removed) // restore vertices in remaining workers
35:
36: end function
```

Algorithm 2 GPSWorker (Greft)

```
1:  function GPSWorker(master)
2:      while master.active() do
3:          msg ← waitMessage(master)
4:          resp ← empty response
5:          if msg.PARTITION then
6:              partition ← msg.partition()
7:          if msg.CREATE_CHECKPOINT then
8:              checkpoint ← partition.createCheckpoint()
9:              resp.addCheckpointHash(checkpoint.computeHash())
10:         if msg.RESTORE_CHECKPOINT then
11:             partition.restoreFromLocalOrReplica(msg.superstep, msg.checkpoint.hashes)
12:             if msg.RESTORE_REPLICAS then
13:                 partition.restoreCheckpointPartition(msg.replicas, msg.checkpoint.hashes)
14:         if msg.START_SUPERSTEP then
15:             for vertex ∈ partition do
16:                 vertex.execute(msg.superstep)
17:             resp.addSuperstepHash(partition.computeHash())
18:         send(resp, master)
19:     if not partition.isReplica then
20:         partition.writeResult()
```

Com a introdução das réplicas de cada processo não há mais necessidade de ar-

mazenar os *checkpoints* num sistema de arquivos distribuídos. Os *checkpoints* passam a ser armazenados na máquina local e, no caso de falha de uma máquina, seu estado pode ser restaurado a partir do último *checkpoint* armazenado em sua réplica (função *restoreFromLocalOrReplica* no algoritmo 2). Esta técnica também é usada na restauração do estado de processos removidos por atingirem f_max ou que falharam por parada. Dessa forma, o sistema de arquivos distribuídos é usado apenas para armazenar os arquivos de entrada e os resultados da computação. Além disso, sempre que um checkpoint é gerado por um *GPSWorker*, um resumo criptográfico do arquivo é enviado ao *GPSMaster*. Com isso, quando for necessário restaurar *checkpoints* o *GPSMaster* devolve estes resumos para os *GPSWorkers*, que podem verificar a integridade de seus arquivos e solicitar o arquivo de sua réplica no caso de uma divergência. Ao final da execução dos *supersteps*, caso o estado dos vértices esteja consistente, apenas uma das réplicas escreve o resultado final no sistema de arquivos distribuídos.

4. Protótipo

O protótipo do Greft foi implementado sobre a versão inicial do GPS disponibilizada pelos autores.[1] O GPS foi escrito em Java, então são descritas as alterações realizadas por classes. Nenhuma alteração foi realizada no HDFS, dado que já existe uma versão tolerante a faltas Bizantinas. Além da implementação dos algoritmos descritos anteriormente, também foi necessária a implementação do armazenamento e restauração de *checkpoints*, uma vez que os mecanismos de tolerância a faltas não foram disponibilizados junto com a versão inicial do GPS.

A classe `GPSMaster` foi alterada com a inclusão dos parâmetros f, f_max e spc. A rotina de particionamento do grafo foi alterada para replicar $f + 1$ vezes os vértices do arquivo de entrada em *GPSWorkers* diferentes. A mensagem de início de um *superstep* foi alterada para incluir os comandos de criação e restauração de *checkpoint*. No laço de controle dos *supersteps*, após o recebimento de todas as mensagens de finalização do *superstep*, foi incluída a comparação dos resumos criptográficos recebidos dos processos. Caso diferenças sejam encontradas, é sinalizada a restauração do último *checkpoint* na mensagem que será enviada para iniciar o próximo *superstep*. Resumos dos arquivos de *checkpoint* de cada processo e sua réplica são incluídos nessa mensagem. Adicionalmente, caso um conjunto de processos tenha ultrapassado o número f_max de diferenças detectadas ou algum processo tenha falhado por parada, eles são removidos. Se existirem máquinas disponíveis, elas são usadas para restaurar os vértices dos processos que foram removidos. Senão, informações sobre os *checkpoints* dos processos faltosos são incluídas nas mensagens enviadas aos processos restantes.

Na classe `GPSWorker`, o laço de controle de *supersteps* foi alterado com a inclusão das rotinas de criação e restauração de *checkpoints* e de geração de resumo criptográfico do estado dos vértices. Ao receber a mensagem de início de um *superstep*, o `GPSWorker` gera ou restaura um *checkpoint* conforme definido na mensagem. Já a geração do resumo criptográfico é feita após o processamento de todos os vértices num *superstep*. O algoritmo SHA-1 é executado sobre o estado de todos os vértices, que estão ordenados numericamente pelo seu identificador. Isso garante que um processo e sua réplica cheguem ao mesmo resumo caso o estado de seus vértices esteja igual. Esse

[1] Obtida em 23/09/2014, no endereço https://subversion.assembla.com/svn/phd-projects/gps/trunk/

resumo e o de um eventual *checkpoint* são adicionados à mensagem de finalização de *superstep* enviada ao `GPSMaster`.

Para as rotinas de criação e restauração de *checkpoints*, foram criadas as classes `CheckpointWriter` e `CheckpointReader`. Elas fazem isso de maneira eficiente, armazenando apenas os dados necessários usando uma codificação binária. A classe `CheckpointReader` obtém o arquivo localmente ou de uma réplica do processo, caso ele esteja corrompido ou ausente. No caso de restauração de *checkpoint* de processos removidos, os dados são carregados para um local temporário e a rotina de distribuição dos vértices usadas no particionamento inicial do grafo é invocada, distribuindo esses vértices entre os processos restantes.

5. Avaliação Experimental

Nesta seção são discutidos os resultados experimentais da execução do protótipo do Greft num *cluster*. Os experimentos foram planejados para avaliar os seguintes aspectos: (1) Qual o custo adicional do algoritmo tolerante a faltas arbitrárias acidentais? (2) Qual o ganho de desempenho obtido com o armazenamento dos *checkpoints* em disco local? (3) Qual o *overhead* introduzido pela recuperação de uma falta?

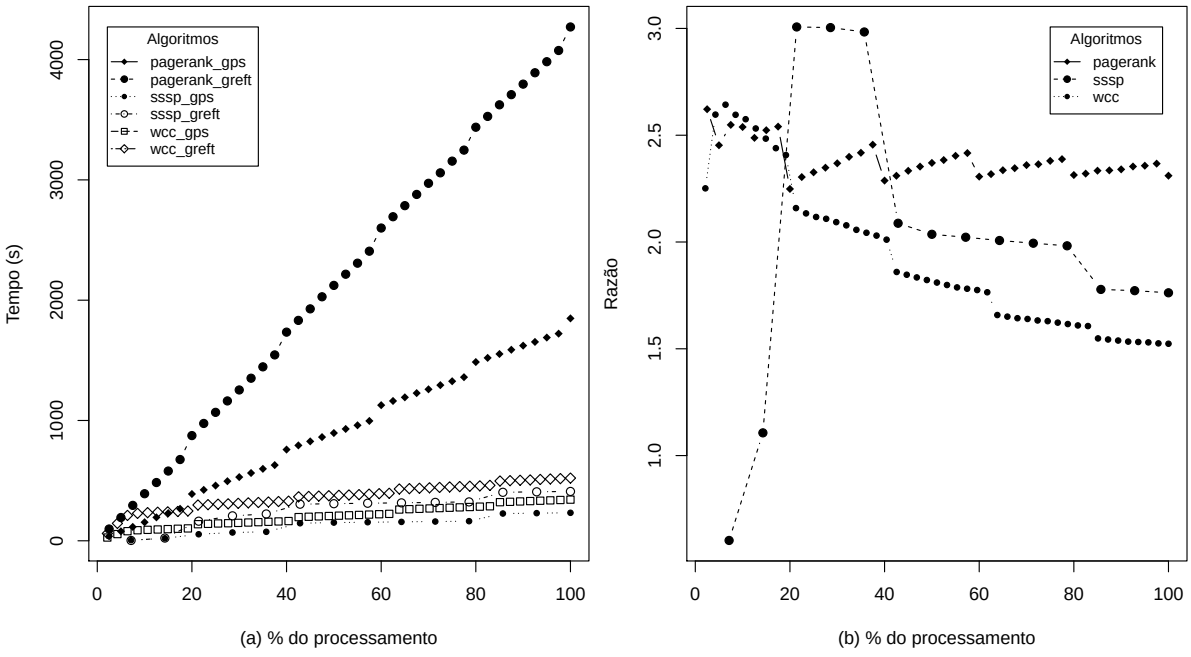


Figure 3. Tempo de execução e razão dos tempos de execução dos algoritmos no GPS original e no Greft

Embora não exista um *benchmark* específico para processamento de grafos, há diversos *datasets* com grafos reais disponíveis. Nos testes foi usado um *dataset* extraído da rede social Twitter, contendo um grafo onde vértices representam usuários e arestas os seus seguidores na rede [Kwak et al. 2010]. Este grafo tem aproximadamente 41,7 milhões de vértices e 1,47 bilhão de arestas. Os experimentos foram executados numa nuvem computacional Amazon Web Services (AWS), usando 17 instâncias (máquinas virtuais) tipo *r3.large* do serviço Elastic Cloud Computing (EC2). Este tipo de instância

possui 15 GB de RAM, 32 GB de armazenamento em SSD e 2 CPUs virtuais de processadores Intel Xeon E5-2670 v2. Todas as instâncias estavam na mesma zona de disponibilidade (*availability zone*) e usavam o Ubuntu Server 14.04 LTS. Como o GPS e o Greft precisam que todo o grafo e as mensagens trocadas fiquem em memória, esse tipo de instância foi escolhido por ter a melhor relação de custo/benefício entre poder de processamento e memória disponível. Além disso, é no próprio ambiente AWS ou similares que aplicações reais de processamento de larga escala são executadas, incluindo processamento de grafos.

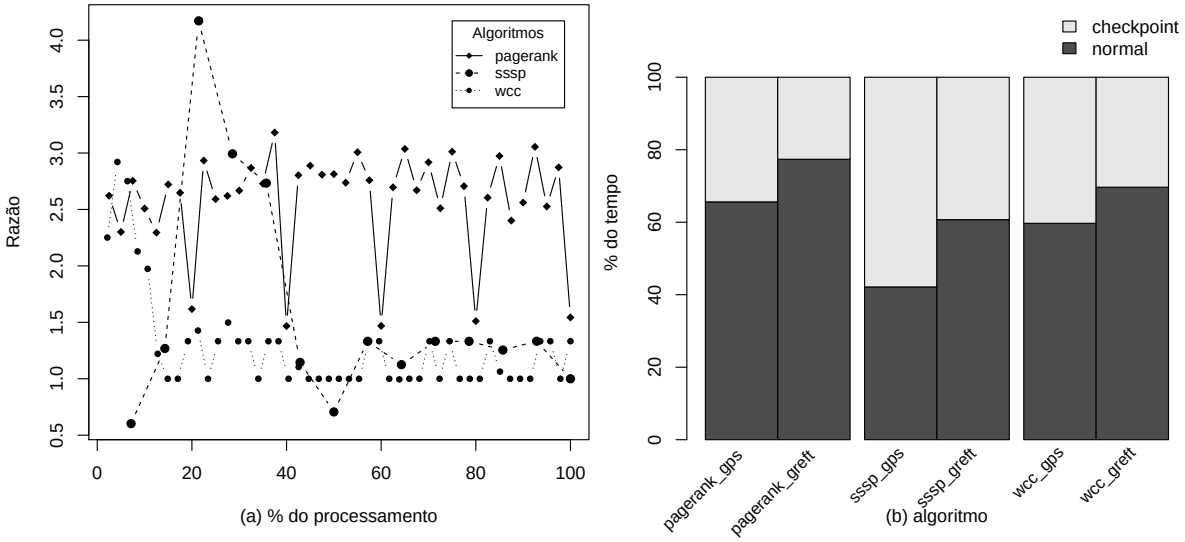


Figure 4. Razão dos tempos de execução dos supersteps e percentuais em supersteps com geração de checkpoints e normais

Três algoritmos foram executados sobre o grafo nos experimentos: *PageRank* - que ranqueia os vértices com base no seu grau de entrada, *Single Source Shortest Path* (SSP) - que calcula a menor distância de um vértice a todos os outros - e *Weak Connected Components* (WCC), que determina as componentes fracamente conexas do grafo. O *PageRank* é caracterizado por um alto número de mensagens trocadas pelos vértices durante todo o processamento e um número fixo de *supersteps* - neste caso, 40. Já o SSP e o WCC têm como característica uma redução do número de vértices ativos com o andamento do processamento, o que faz que o volume de mensagens trocadas entre os vértices e o processamento propriamente dito diminuam com o tempo. O número de *supersteps* necessários para completar o algoritmo depende da estrutura do grafo. No grafo usado nos experimentos, o SSP finalizou com de 17 *supersteps*, e o WCC com 47.

Nos experimentos o algoritmo foi parametrizado com $f = 1$, pois esse é o valor usualmente assumido em testes de sistemas tolerantes a faltas Bizantinas e porque é pouco provável ter um número de faltas superior a esse num passo de processamento. Além disso apenas faltas transientes foram simuladas, pois a implementação da remoção de uma réplica do sistema não estava completa na versão disponível do GPS original. O parâmetro *spc* foi definido conforme as características de cada algoritmo e ficou em 8 para o PageRank, 6 para o SSP e 10 para o WCC. Levou-se em consideração apenas o tempo de execução dos *supersteps* de cada algoritmo, sem considerar o tempo para particionar o

grafo entre as réplicas no início e de armazenamento do resultado no final do algoritmo. Embora o particionamento inicial do grafo seja afetado pela implementação do novo algoritmo, o fator dominante é a forma como os arquivos de entrada estão distribuídos no HDFS, cuja otimização foge do escopo desse artigo. Já a escrita final do resultado não foi afetada pelas implementações, pois apenas uma réplica armazena o resultado. Os valores exibidos são a média de 3 execuções de cada algoritmo.

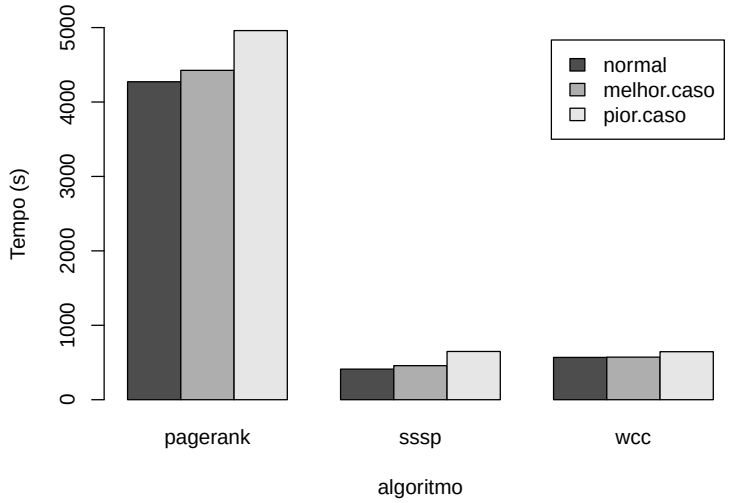


Figure 5. Tempo de execução dos algoritmos no GPS original e no Greft

O primeiro experimento foi a execução dos três algoritmos sobre o grafo do Twitter, tanto no modo original do GPS quanto no modo tolerante a faltas arbitrárias acidentais do Greft, para comparação do desempenho. Os tempos médios de execução (e desvios-padrão) para o PageRank no GPS e Greft foram 1849 ($\pm$ 62) e 4272 ($\pm$ 119) segundos; para o SSSP, 233 ($\pm$ 13) e 411 ($\pm$ 21) segundos; para o WCC, 342 ($\pm$ 11) e 568 ($\pm$ 17) segundos. A figura 3 mostra (a) estes tempos de execução e (b) a razão do tempo de processamento do Greft e da versão original do GPS (ou seja, tempo do Greft dividido pelo tempo do GPS). Nota-se que a versão experimental leva, em média, duas vezes o tempo da versão oficial, variando conforme o algoritmo. Isso é esperado, já que o Greft tem à disposição metade dos recursos da versão original devido à replicação dos vértices.

Uma das otimizações do novo algoritmo é o armazenamento dos *checkpoints* em disco local. Esperava-se um ganho no desempenho por se dispensar o uso do HDFS, mesmo considerando que os *checkpoints* no Greft teriam o dobro do tamanho dos do GPS original. Esse ganho foi alcançado, como pode ser visto na figura 4. Neste gráfico é exibida (a) a razão do tempo de execução entre o Greft e GPS original de cada *superstep* dos algoritmos e (b) o percentual de tempo gasto em *supersteps* com geração de *checkpoints* e *supersteps* normais. Os vales visíveis no gráfico (a) ocorrem nos *supersteps* onde há geração de um *checkpoint*, e demonstram que o desempenho do Greft fica próximo do GPS original nestes pontos. No gráfico (b) percebe-se que o percentual do tempo gasto em *supersteps* que geram *checkpoints* é menor no Greft.

Outro experimento realizado foi a execução dos algoritmos com a injeção de uma falta transiente, para avaliar o *overhead* introduzido pela rotina de recuperação de faltas (figura 5). Foram simulados os cenários de melhor e pior caso. Assumindo que o fator

dominante na restauração de uma falta é o número de *supersteps* que serão reexecutados após a restauração do *checkpoint*, o melhor caso é uma falta que ocorre logo após um *superstep* onde um *checkpoint* foi gerado, pois apenas um *superstep* precisará ser reexecutado. Já o pior caso é o de uma falta que ocorra imediatamente antes da geração de um *checkpoint*. Na figura, é possível perceber que nos melhores casos o *overhead* é realmente muito pequeno - limitando-se ao tempo de restauração do estado e de execução de mais um *superstep*. Já no pior caso, há um *overhead* maior, mas também limitado ao número de *supersteps* que precisaram ser reexecutados após a restauração do *checkpoint*

6. Trabalhos Correlatos

Existe uma vasta literatura sobre tolerância a faltas Bizantinas ou arbitrárias, com modelos propostos desde a década de 80 [Lamport et al. 1982]. Replicação de máquinas de estado é uma técnica genérica para lidar com faltas, que já se demonstrou poder ser implementada de forma eficiente mesmo tolerando faltas Bizantinas [Castro and Liskov 2002]. Depois desse trabalho, vários algoritmos eficientes apareceram, como a biblioteca UpRight [Clement et al. 2009] e a EBAWA [Veronese et al. 2010]. Modelos para processamento paralelo de larga escala, como o *MapReduce*, também contam com propostas para tolerar faltas Bizantinas, como em [Costa et al. 2011] e [Stephen and Eugster 2013]. Entretanto, como já discutido, estes modelos não são adequados ao processamento distribuído de grafos, principalmente pelo custo envolvido nestas técnicas.

As características dos sistemas mais próximos deste trabalho são brevemente descritas na tabela 1. Pregel [Malewicz et al. 2010], em que é baseado o GPS, é um dos primeiros modelos e introduziu vários dos conceitos usados em outros trabalhos, como a definição do *superstep*. Entretanto, ele tolera apenas faltas de parada (*crash*) através de um mecanismo de recuperação em retrocesso baseado em *checkpoints*. PowerGraph [Gonzalez et al. 2012] e Trinity [Shao et al. 2013] usam mecanismos semelhantes e também toleram apenas faltas de parada. Graphx [Xin et al. 2013] implementa tolerância a faltas de parada estendendo o mecanismo *Resilient Distributed Datasets* (RDD), presente no *framework Spark*, que foi usado para construí-lo. Este mecanismo armazena as alterações realizadas nos *datasets* em vez de armazenar os dados propriamente ditos. Nenhum desses modelos usa replicação para implementar tolerância a faltas.

	Mecanismo de T.F.	**Tipo de falta**	**Réplicas**
Pregel/GPS	Recuperação em retrocesso	crash	-
PowerGraph	Recuperação em retrocesso	crash	-
Trinity	Recuperação em retrocesso	crash	-
GraphX	RDD	crash	-
Imitator	Replicação	crash	$f+1$
Greft	*Replicação*	*arbitrárias acidentais*	$f+1$

Table 1. Trabalhos relacionados e suas características

Imitator [Wang et al. 2014], baseado no Pregel, propõe a replicação de vértices em memória para tolerar faltas de parada, dispensando o uso de *checkpoints*. Para isso, o *Imitator* se utiliza de uma técnica existente em alguns sistemas de processamento de grafos, que consiste em replicar vértices com grau muito alto em processos diferentes para tornar o acesso a eles mais rápido nestes processos. Estendendo essa técnica, o *Imitator*

cria réplicas de todos os vértices do grafo. No caso de falha de um processo, essas réplicas são redistribuídas entre os processos restantes para dar continuidade ao processamento.

7. Conclusão

Neste artigo foi apresentado o Greft, um algoritmo e protótipo de um sistema de processamento distribuído de grafos tolerante a faltas arbitrárias acidentais. Também foi apresentada uma avaliação experimental do protótipo, onde se confirmou que ele é capaz de lidar com este tipo de faltas usando, no máximo, duas vezes os recursos do modelo original, usando o parâmetro $f = 1$. Este valor é realista, já que este tipo de falta não acontece com frequência e, quando acontece, tem uma probabilidade muito baixa de deixar duas réplicas com o mesmo estado. As otimizações realizadas no armazenamento dos *checkpoints* também se mostraram satisfatórias, reduzindo significativamente o tempo gasto nesta tarefa quando comparada ao modelo original.

O modelo também se mostrou eficiente ao se recuperar de faltas transientes, adicionando um *overhead* referente apenas à restauração do estado e reexecução dos *supersteps* que já haviam sido processados desde o último *checkpoint*. Isso é um avanço se comparado ao modelo original, onde seria necessário executar o processamento três vezes para chegar ao mesmo resultado no caso de uma falta transiente.

References

Avery, C. (2011). Giraph: Large-scale graph processing infrastructure on Hadoop. In *Proceedings of Hadoop Summit*.

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

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 Riche, T. (2009). UpRight cluster services. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*, pages 277–290.

Costa, P., Pasin, M., Bessani, A. N., and Correia, M. (2011). Byzantine fault-tolerant MapReduce: Faults are not just crashes. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing Technology and Science*, pages 32–39.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Driscoll, K., Hall, B., Sivencrona, H., and Zumsteg, P. (2003). Byzantine fault tolerance, from theory to reality. In *Computer Safety, Reliability, and Security*, pages 235–248. Springer.

Gonzalez, J. E., Low, Y., Gu, H., Bickson, D., and Guestrin, C. (2012). PowerGraph: Distributed graph-parallel computation on natural graphs. In *Proceedings of the 10th Symposium on Operating System Design and Implementation*.

Kajdanowicz, T., Kazienko, P., and Indyk, W. (2014). Parallel processing of large graphs. *Future Generation Computer Systems*, 32:324–337.

Kwak, H., Lee, C., Park, H., and Moon, S. (2010). What is Twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, pages 591–600.

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

Malewicz, G., Austern, M. H., Bik, A. J., 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.

Nightingale, E. B., Douceur, J. R., and Orgovan, V. (2011). Cycles, cells and platters: an empirical analysis of hardware failures on a million consumer pcs. In *Proceedings of the EuroSys 2011 Conference*, pages 343–356.

Salihoglu, S. and Widom, J. (2013). GPS: A graph processing system. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*.

Schroeder, B. and Gibson, G. A. (2007). Understanding failures in petascale computers. In *Journal of Physics: Conference Series*, volume 78.

Schroeder, B., Pinheiro, E., and Weber, W.-D. (2009). DRAM errors in the wild: a large-scale field study. In *ACM SIGMETRICS Performance Evaluation Review*, volume 37, pages 193–204.

Shao, B., Wang, H., and Li, Y. (2013). Trinity: A distributed graph engine on a memory cloud. In *Proceedings of the 2013 International Conference on Management of Data*, pages 505–516.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The Hadoop distributed file system. In *IEEE 26th Symposium on Mass Storage Systems and Technologies*, pages 1–10.

Stephen, J. J. and Eugster, P. (2013). Assured cloud-based data analysis with ClusterBFT. In *Middleware 2013*, pages 82–102. Springer.

Valiant, L. G. (1990). A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111.

Veronese, G. S., Correia, M., Bessani, A. N., and Lung, L. C. (2010). EBAWA: Efficient Byzantine agreement for wide-area networks. In *IEEE 12th International Symposium on High-Assurance Systems Engineering*, pages 10–19.

Wang, P., Zhang, K., Chen, R., Chen, H., and Guan, H. (2014). Replication-based fault-tolerance for large-scale graph processing. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 562–573.

Xin, R. S., Gonzalez, J. E., Franklin, M. J., and Stoica, I. (2013). GraphX: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*.

Federação de *Clouds* e Atributos de Segurança

Luciano Barreto, Joni da Silva Fraga, Frank Siqueira

Universidade Federal de Santa Catarina
Caixa Postal 476 – 88040-900 – Florianópolis – Santa Catarina – Brasil

{lucianobarreto, fraga}@das.ufsc.br, frank@inf.ufsc.br

Resumo. *Atualmente, o conceito de Federações tem atraído a atenção da comunidade de pesquisa na área de computação em nuvem. Em uma federação de clouds, provedores de nuvem criam relações de confiança e compartilham recursos, seja por escassez momentânea ou com o intuito de oferecer uma maior gama de recursos. Este artigo apresenta uma infraestrutura para formação de uma federação de clouds, que define a forma de organização e estratégias para busca e aquisição de recursos na federação. Além disso, de modo a permitir o gerenciamento de identidades entre domínios, a infraestrutura proposta incorpora mecanismos de autenticação e autorização no contexto da federação. Resultados de simulações mostram a viabilidade da infraestrutura proposta.*

Abstract. *Nowadays, the concept of Federations has drawn attention of the research community in the area of cloud computing. In a cloud federation, cloud providers build trust relationships and share resources, aiming to overcome momentary resource shortage or to offer a wider range of resources to their users. This paper introduces an infrastructure for building a cloud federation, which defines organization principles and strategies for searching and acquiring resources in a federation. In addition, aiming to allow cross-domain identity management, the proposed infrastructure incorporates authentication and authorization mechanisms in the context of a federation. Simulation results show the feasibility of the proposed infrastructure.*

1. Introdução

Nos últimos anos, a computação em nuvem tem sido foco de grande parte dos esforços de pesquisa da comunidade de sistemas distribuídos. Muitas das noções pré-estabelecidas de sistemas e de aplicações distribuídas vêm sendo revistas diante dos requisitos postos por estes novos e complexos ambientes de processamento distribuído.

As grandes demandas que esses sistemas devem suportar têm feito dos mesmos foco de experimentações e de busca de novos conceitos, objetivando manter bons níveis de aproveitamento de recursos com alta qualidade de serviço. O conceito de *federação*, que vem sendo amplamente empregado no sentido de facilitar o gerenciamento de identidades, começa a ser explorado na literatura de *clouds*, visando a cooperação entre diferentes provedores. A integração em federações permite que provedores de *clouds* colaborem de forma a compartilhar recursos no atendimento a demandas de usuários. Os tipos de compartilhamento podem ocorrer em diversos níveis de recursos (máquinas virtuais, armazenamento, redes virtuais, *software*, etc.) e em diferentes situações, tais como: a falta de recursos em um provedor para suportar grandes demandas, o fornecimento de serviços

ou recursos originalmente não oferecidos por um provedor, a distribuição de uma aplicação em diferentes provedores (definida pelo próprio usuário), em caso de colaboração entre pequenos provedores de *clouds*, dentre outros.

O interesse atual na investigação do uso do conceito de federação de *clouds* é, portanto, justificável. No entanto, o agrupamento de provedores de *clouds* formando redes de confiança que atendem um vasto número de usuários coloca também grandes desafios na autenticação e autorização destes usuários junto aos provedores. Para contornar possíveis dificuldades no trato de aspectos de segurança, modelos de gerenciamento de identidades se tornam importantes para esses grandes sistemas.

O foco principal deste trabalho é a apresentação de uma proposta de modelo de federação de *clouds* no qual são identificadas as principais entidades que suportam o conceito de federação. O modelo a ser proposto descreve ainda os protocolos usados na busca e aquisição de recursos na federação de *clouds*, em atendimento às demandas de usuários. Estes protocolos comportam também mecanismos que visam atender requisitos de segurança, como os controles de autenticação e de autorização na federação de *clouds*.

Os serviços de federação são fornecidos a partir de uma infraestrutura de serviços e de um provedor de identidades. As diversas entidades (provedores e usuários) da federação confiam nos serviços dessa infraestrutura e nas autenticações do provedor de identidades, o que permite que provedores de *clouds*, individualmente, não se preocupem em manter extensas bases de dados com identidades e atributos de usuários. Neste texto, além de apresentarmos o modelo proposto e avaliarmos os resultados obtidos através de simulações, confrontamos as características funcionais do modelo com as propostas relacionadas presentes na literatura.

2. Federação de Provedores de *Cloud*

Em [Grozev and Buyya 2012], o conceito de federação de *clouds* é apresentado como um conjunto de provedores que interconectam suas infraestruturas com o objetivo de compartilhar recursos. Porém, além do termo federação, outros termos são usados para nomear a cooperação de provedores no compartilhamento de recursos. Entre outros termos, são citados na literatura: *Inter-cloud* [Grozev and Buyya 2012], *Multi-cloud* [Grozev and Buyya 2012] e *Cross-Cloud* [Celesti et al. 2010]. Embora as diferenças de nomenclatura e mesmo que muitas das propostas para a formação de redes de confiança de *clouds* tenham diferentes focos, o objetivo final é sempre o uso de serviços de diversos provedores de *clouds* para atender a demanda e o *SLA* (*Service-Level Agreement*) de cliente.

2.1 Proposta de Modelo de Federação

O modelo introduzido neste artigo foi desenvolvido para que usuários, diante da necessidade de recursos computacionais, solicitem seus *SLAs* para que uma entidade de suporte, baseada nas informações de demanda, busque por recursos na federação de *clouds* que atendam a requisição do usuário. Com isso, os usuários não se preocupam em como ou onde os recursos estão disponíveis para seu uso. A localização dos recursos é sempre controlada por uma infraestrutura global de serviços que suporta a federação de *clouds*.
Uma visão geral do modelo proposto é ilustrada pela Figura 1, na qual as diferentes entidades do modelo são representadas. A primeira destas corresponde a uma *API* denominada de *Cloud User*, que permite aos usuários interagir com a federação, enviando *SLAs* e requisições de recursos.

As interações do usuário com a federação no modelo proposto são feitas através de uma infraestrutura chamada de Suporte da Federação (*SF*), que é responsável por receber as requisições de recursos e instanciar *brokers* para negociar e controlar diretamente o *SLA* de seu usuário na federação de *clouds*. Para cada *SLA* de usuário, um *broker* é instanciado e permanece ativo até o fim deste contrato. Cada *broker* atua em nome do seu usuário na aquisição dos recursos, entregando *tokens* de autenticação aos provedores de *cloud* de forma a comprovar estes requisitantes como usuários válidos na federação.

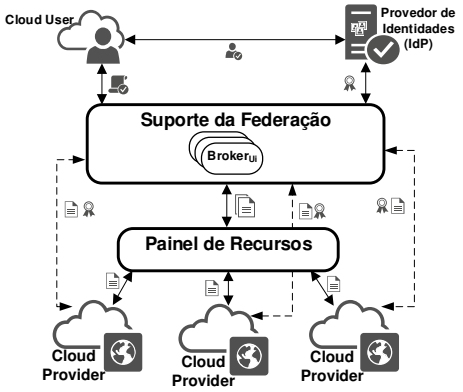


Figura 1 – Visão geral da proposta de federação

O modelo pressupõe o envolvimento de diversos provedores de *cloud* (*Cloud Providers*) oferecendo diferentes tipos de serviço e zelando única e exclusivamente pelos seus próprios recursos. Estes provedores interagem com os *brokers* no momento da busca e aquisição dos recursos. Após este processo, os recursos considerados serão mantidos disponíveis aos usuários envolvidos. Os provedores de *cloud* mantêm seus próprios controles de segurança (autorização), de forma que somente ofereçam recursos para usuários que tenham a posse de um *token* de autenticação.

Nas requisições de recursos, as interações entre *brokers* e provedores de *cloud* no modelo proposto são concretizadas através do componente Painel de Recursos (*PR*), que se comporta de forma semelhante a um serviço de eventos. Para receberem pedidos vindos de *brokers*, provedores devem se registrar no Painel como receptores de notificações adequadas a seus recursos disponíveis. No Painel de Recursos são mantidas listas de provedores ativos e registrados, agrupados pelo tipo de recurso que tornam disponível na federação. No recebimento de demandas, o Painel verifica quais são os recursos solicitados e notifica diretamente os provedores. Os provedores então respondem ao *broker* correspondente informando sua disponibilidade dos recursos solicitados.

Um Provedor de Identidades (*IdP*) foi introduzido no modelo com funções de manter, gerenciar e autenticar usuários e seus atributos na federação. Cada usuário, no acesso ao Suporte da Federação (*SF*), deve inicialmente se autenticar junto a este *IdP*. Autenticidades verificadas geram *tokens* (ou asserções) de autenticação, que devem garantir aos usuários validados a busca e o uso de recursos nos diversos provedores de *cloud* da federação. Os provedores de *cloud* confiam nas autenticações de usuários do *IdP* e recursos só são liberados para usuários previamente autenticados na federação.

2.2 Componentes e Protocolos

As próximas subseções apresentam os algoritmos que descrevem como são feitas as trocas na busca e alocação de recursos entre as entidades do modelo. Estas buscas e acessos

aos recursos são acompanhados por controles de autenticação e de autorização dos usuários da federação.

2.2.1 *Cloud User*

O Algoritmo 1 apresenta o processo de solicitação de recursos na federação, onde o usuário inicialmente gera seu *SLA* (linhas 13 – 17), especificando os parâmetros de contrato (preço, *uptime* mínimo, qualidade mínima de serviço, dentre outros). Os tipos de cada recurso desejado são selecionados (linhas 19 – 22, Algoritmo 1) na sequência. A requisição $fedReq_i$ é montada (linha 25), armazenada (linha 26) e por fim remetida ao *SF* (linhas 27). Uma vez enviada a requisição, em um primeiro momento o usuário deve receber um pedido de autenticação ($authCReq_i$) do *IdP*. O usuário então responde ao *IdP* com suas credenciais (usuário/senha, certificado, etc.) através da mensagem $authCred_i$ (linhas 28–30). Após esse passo, o usuário aguarda pelos ponteiros (endereços) dos recursos, os quais serão enviados pelo *SF*, por meio do *broker* alocado para o seu *SLA*, em uma mensagem do tipo $fedRep_i$ (linhas 31 e 32).

Algoritmo 1 – *Cloud User*

Algoritmo 1. *Cloud User* U_i

```
Structures
1:    Resource {
2:      Type;
3:      Amount;}
4:  Type:
5:    VmTypeA = {1ghz cpu, 1gb mem, ...}
6:    VmTypeB = {2 ghz cpu, 2 gb mem, ...}
7:  Init:
8:    n_u ← 0;
9:    fedReqList[] ← Ø;
10:   res[] ← Ø;

11:   upon ResourceRequest do
12:     //Create SLA requirements. Example:
13:     price ← ≥ X;
14:     uptime ← ≤ 99%;
15:     QoS ←≥Y;
16:     ....
17:     sla_i ← <SLA, price, uptime, QoS, ...>
18:     //Create resource requirements. Example:
19:     res ← res ∪ Resource(VmTypeA, 2);
20:     res ← res ∪ Resource(VmTypeB, 3);
21:     …
22:     resReq_i ← <RESREQ, res>;
23:     n_u++;
24:     id_req ← <IDREQ, id_Ui, n_u>;
25:     fedReq_i ← <FEDREQ, id_Ui, id_req, sla_i, resReq_i, sign_ui, cert_ui>
26:     fedReqList ← fedReqList ∪ fedReq_i;
27:     send fedReq_i to SF

28:   upon Receive authCReq_i from IdP do
29:     authCred_i ←<AUTHCRED_Ui, id_Ui, id_req, credentials, sign_ui, cert_ui>;
30:     send authCred_i to IdP

31:   upon Receive fedRep_i from B_i do
32:       resourceRequested ← fedRep_i.fedAlRes;
```

Todas as mensagens trocadas entre as entidades são assinadas de modo a garantir sua a autenticidade e integridade. Ou seja, as entidades do modelo só trocam informações com outras entidades credenciadas e devidamente validadas. Por exemplo, com a mensagem $fedReq_i$ (linha 25) é enviada a sua assinatura $sign_{ui}$ e o certificado de chave pública do assinante ($cert_{ui}$) para validação posterior da mensagem.

2.2.2 Suporte da Federação

A entidade identificada como Suporte da Federação (*SF*) é responsável por receber requisições e instanciar *brokers* para atender as solicitações de usuários. O algoritmo 2 descreve seu comportamento nas interações com o *Cloud User*. Neste algoritmo, ao receber uma requisição $fedReq_i$ (linha 3), o *SF* armazena e recupera algumas informações importantes, como por exemplo, o identificador do usuário (linhas 4 – 6, Algoritmo 2).

De posse do identificador id_{ui} do usuário, o *SF* executa um procedimento de descoberta [Miller 2006] (linha 7) que visa encontrar o *IdP* que armazena as credencias do usuário. Executado esse protocolo de descoberta, o Suporte da Federação (*SF*) envia então uma requisição de autenticação ($authReq_i$) ao *IdP* correspondente (linhas 8 e 9). Após esse passo de autenticação, o *SF* aguarda um *token* de autenticação ($authToken_i$) vindo do *IdP* que confirme o emissor da requisição como usuário válido. Uma vez recebido esse

token, o *SF* faz as verificações necessárias da assinatura do *token* que, caso sua validade seja confirmada, instancia um *broker* para atender a requisição do usuário (linhas 10 -14).

Algoritmo 2 – Suporte da Federação

Algoritmo 2. *Suporte da Federação F*

1. **Init:**
2. $fedReqList \leftarrow \emptyset$;
3. **upon** ***Receive*** $fedReq_i$ **from** U_i **do**
4. $fedReqList \leftarrow fedReqList \cup fedReq_i$;
5. $id_{req} \leftarrow fedReq_i.id_{req}$;
6. $id_{ui} \leftarrow fedReq_i.id_{Ui}$;
7. $IdP \leftarrow$ ***DiscoverIDP***(id_{ui});
8. $authReq_i \leftarrow <AUTHREQ, id_{ui}, id_{req}, sign_F, cert_F>$;
9. ***send*** $authReqi$ ***to*** IdP;
10. **upon** ***Receive*** $authToken_i$ **from** IdP
11. $token_{ui} \leftarrow authToken_{Ui}.token_{ui}$;
12. $id_{req} \leftarrow authToken_i.id_{req}$;
13. **if** ***verifyToken***$(token_{ui}) \wedge fedReqList.get(id_{req})$ **then**
14. $broker_{Ui} \leftarrow$ ***createBroker***$(fedReq_i, token_{ui})$;
15. **end if;**

2.2.3 *Broker*

O *broker* é o agente instanciado com o objetivo de efetuar a negociação e alocação de recursos na federação. O algoritmo 3 descreve o comportamento do *broker* na busca e gerência de um *SLA* de usuário. Para a instanciação de um *broker* (linha 8 - Algoritmo 3), o *SF* passa como parâmetro a requisição do usuário assim como o *token* gerado pelo *IdP* no processo de autenticação. Esse *token* será usado posteriormente nos controles de autorização quando da requisição de recursos em provedores de *cloud* da federação.

O próximo passo desse *broker* recém-instanciado é então encaminhar a requisição $fedReq_iB_i$, em nome de seu usuário, ao Painel de Recursos (*PR*) (linha 14). Estas mensagens incorporam o identificador do *broker* e a requisição do usuário. Vale ressaltar que a assinatura desta mensagem é feita com a chave privada do Suporte da Federação (*SF*) e é o seu certificado que vai com esta requisição de modo a validar esta assinatura. A requisição $fedReq_iB_i$, com as solicitações do usuário, é então enviada pelo *broker* ao *PR* (linha 15). Como consequência, um temporizador é iniciado pelo *broker* para controlar em prazo estipulado as recepções de respostas a esta requisição (linha 16). No caso de decorrência deste prazo (linhas 44 – 47), a mensagem $fedReq_iB_i$ é enviada novamente ao *PR* e um novo prazo é estimado para o recebimento das respostas correspondentes.

No recebimento destas mensagens de resposta ($fedRes_i$), o *broker* verifica se o provedor correspondente pode atender totalmente os recursos solicitados na requisição do usuário (linhas 17-19). Em caso afirmativo, o temporizador iniciado para espera de respostas é removido e uma resposta $fedReq_iAck_i$ é enviada a este provedor (linhas 21-23), informando a concordância em alocar todos os recursos indicados. Caso a resposta do provedor não indique o atendimento total da requisição do usuário, o *broker* então armazena essa resposta para uma possível alocação distribuída entre os diversos provedores de *cloud* da federação (linhas 24 - 26). Quando a soma dos recursos que podem ser suportados parcialmente pelos provedores de *cloud* for suficiente para o atendimento das necessidades do usuário (linha 27), o *broker* então remove o temporizador (linha 28) e ordena a lista de provedores que atenderam a solicitação de recursos via Painel (linha 29). Esta ordenação usa o critério de número máximo de recursos livres por provedor de *cloud*, tentando com isto minimizar o número de provedores que atendam a requisição do usuário.

Efetuado o envio da confirmação para que os recursos sejam alocados, o *broker* então fica no aguardo de mensagens $fedAllocAck_i$. Nessas mensagens são enviados os ponteiros que indicam onde os recursos estão disponíveis. Com o recebimento dessas mensagens $fedAllocAck_i$ (linha 37 - Algoritmo 3), o *broker* armazena os ponteiros em uma

lista e verifica se o número de ponteiros recebidos atende ao número de recursos da requisição do cliente (linha 40). Caso a verificação seja verdadeira, uma mensagem $fedRep_i$ é gerada com os ponteiros dos recursos (linha 41) e enviada para o *Cloud User* (linha 42, Algoritmo 3), finalizando o processo de aquisição de recursos.

Algoritmo 3 - *Broker*

Algoritmo 3. *Cloud Broker* B_i

```
1:  Init:
2:  resReq ← Ø;
3:  partFedRes ← Ø;
4:  authTkn_Ui ← Ø;
5:  fedAlRes ← Ø;
6:  timeout ← initialValue;
7:  δ_B ← Ø
8:  upon CreateBroker(fedReq_i, token_Ui) do
9:    idU ← fedReq_i.id_req.id_ui;
10:   id_Bi ← <IDBI, B_i, idU>;
11:   id_req ← fedReq_i.id_req;
12:   resReq ← fedReq_i.resReq_i.res;
13:   authTkn_Ui ← tokenUi;
14:   fedReq_iB_i ← <FEDRB, id_Bi, id_req, fedReq_i, sign_F, cert_F>;
15:   send fedReq_iB_i to RP;
16:   δ_B.time ← time();
17: upon Receive fedRes_i from C_j do
18:   id_req ← fedRes_i.id_req;
19:   if fedRes_i.canff = "Full" then
20:     nRes ← "Full";
21:     δ_B.time ← ⊥;
22:     fedReq_iAck_i ← <FEDREQACK, id_Bi id_req, fedRes_i, ↩
            Full, sign_F, cert_F>;
23:     send fedReq_iAck_i to C_j;
24:   else
25:     partFedRes ← partFedRes ∪ fedRes_i;
26:     fedResCount ← fedResCount + fedRes_i.MaxFreeRes;
27:     if (fedResCount ≥ | resReq|) then
28:       δ_B.time ← ⊥;
29:       orderByMaxFreeResources(partFedRes);
30:       for all C_j∈ partFedRes
31:         nRes = [i…n];
32:         fedReq_iAck_i ← <FEDREQACK, id_Bi, id_req, ↩
              fedRes_i, nRes, authTkn_Ui, sign_F, cert_F>;
33:         send fedReq_iAck_i to C_j;
34:       end for;
35:     end if;
36: end if;
37: upon Receive fedAllocAck_i do
38:   id_req ← fedAllocAck_i.id_req;
39:   fedAlRes ← fedAlRes ∪ fedAllocAck_j.resPointers;
40:   if (|fedAlRes| ≥ |resReq|) then
41:     fedRep_i ← <FEDREPLY, id_Bi,id_req, ↩
            fedAllocRes, sign_F, cert_F>;
42:     send fedRep_i to U_i;
43:   end if;
44: upon (time() - δ_B.time) ≥ timeout do
45:   send fedReq_iB_i to RP;
46:   δ_B.time ← time();
47:   timeout ← NewEstimation(timeout);
```

2.2.4 Painel de Recursos

Como citado anteriormente, um serviço de eventos especializado, nomeado Painel de Recursos (*PR*), integra o modelo de federação proposto, permitindo com isso que os provedores de *cloud* sejam notificados sobre demandas de recursos. O comportamento do *PR* é explicitado no Algoritmo 4. Ao receber uma requisição $fedReq_iB_i$ (linha 8, Algoritmo 4), esse serviço (*PR*) deve notificar aos provedores apropriados sobre a demanda correspondente. O *PR* percorre então as listas de provedores (linhas 12 - 23), enviando mensagens $reqNotif_i$ para todos os provedores que oferecem determinados tipos de recursos.

Diante da disponibilidade ou indisponibilidade momentânea de recursos, os provedores de *cloud* da federação podem se inscrever ou pedir para ser removidos das listas associadas a tipos de recursos. Ou seja, essas listas evoluem refletindo a dinamicidade da federação em termos da disponibilidade de recursos. O *PR* oferece duas operações em sua interface que permitem que provedores entrem ou saiam das listas associadas: $memEnter_m$ e $memExit_m$. O *PR*, ao receber uma mensagem $memEnter_m$, adiciona o provedor em listas (associadas) de tipos de recursos que este provedor dispõe, deixado o mesmo habilitado para receber notificações correspondentes (linhas 25 - 32). No recebimento de uma mensagem $memExit_m$, o *PR* remove o provedor das listas associadas de tipos de recursos que o mesmo já não dispõe (linhas 34 – 40).

2.2.5 *Cloud Provider*

Provedores de *cloud* são os fornecedores de recursos na federação. Uma vez estabelecidos os contratos (via *brokers*), os usuários passam a ter acesso direto aos recursos alocados, segundo os *SLAs* estabelecidos.

Algoritmo 4 – Painel de Recursos

Algoritmo 4. *ResourcePanel RP*

```
1.  Structures
2.    Member{
3.      Type} // VmTypeA, VmTypeB
4.      Name[]}; //C1, C2, C3....
5.  Init:
6.  memList ← Members;
7.  reqList ← Ø;

8.    upon Receive fedReqiBi from Bi
9.      idreq ← fedReqiBi.idreq;
10.     idRP ← <IDRP, idRP, idReqP>;
11.     reqList ← reqList ∪ fedReqiBi;
12.     for all Type ∈ fedReqiBi.resReqi.res.Type do
13.       case Type is
14.         VmTypeA:
15.             type ← VMTypeA;
16.             reqNotifi ← <REQNOTIF, idRP, idreq, fedReqiF, ↩
                    type, signrp, certrp>;
17.             send reqNotifi to all Cj ∈ MemList.Type(VmTypeA);
18.         VmTypeB:
19.             type ← VMTypeB;
20.             reqNotifi ← <REQNOTIF, idRP, idreq, fedReqiF, ↩
                    type, signrp, certrp>;
21.             send reqNotifi to all Cj ∈ memList.Type(VmTypeB);
22.       ...
23.     end case
24.   end for

25.   upon Receive memEnterm from Cj
26.     case memEnterj.Type is
27.       VmTypeA:
28.         memList.VmTypeA ←memList.VmTypeA ∪ memEnterm.Cj;
29.       VmTypeB:
30.         memList.VmTypeB ←memList.VmTypeA ∪ memEnterm.Cj;
31.       ...
32.     end case

33.   upon Receive memExitm from Cj
34.     case memExitj.Type is
35.       VmTypeA:
36.         memList.VmTypeA ←memList.VmTypeA - memEnterm.Cj;
37.       VmTypeB:
38.         memList.VmTypeB ←memList.VmTypeB - memEnterm.Cj;
39.       ....
40.     end case
```

Ao receber uma notificação (*reqNotif*$_i$) do Painel de Recursos, o provedor de *cloud* primeiramente recupera o contrato de *SLA* da requisição do usuário e verifica se o mesmo está de acordo com os parâmetros e requisitos definidos para o fornecimento de recursos (linhas 9-14, Algoritmo 5). Caso o provedor possa atender ao *SLA*, a disponibilidade dos recursos solicitados é verificada (linha 15).

O atendimento pelo provedor de uma solicitação de recursos pode ocorrer de duas maneiras: de forma total, onde o provedor pode atender toda a demanda do usuário, ou parcial, quando somente parte dos recursos requisitados estão disponíveis. Se a quantidade de recursos no provedor é suficiente para atender por completo a solicitação (linha 16), o provedor de *cloud* faz então a reserva desses recursos (linha 17) e monta a mensagem *fedRes*$_i$ que informa o atendimento integral (*canff* ← *Full*) pelo provedor da solicitação do usuário. Essa mensagem de resposta à notificação *reqNotif*$_i$ é enviada diretamente ao *broker* solicitante (linhas 19-20). Um temporizador é iniciado após o envio da mensagem *fedRes*$_i$ (linha 21) para controlar um prazo definido segundo políticas do provedor de *cloud*. Este prazo é usado para liberar os recursos reservados, caso não haja resposta de confirmação do *broker* dentro do limite de tempo estipulado (linhas 45 - 46).

No caso da impossibilidade de atendimento total da requisição do usuário, o provedor monta sua mensagem *fedRes*$_i$, informando o atendimento parcial da solicitação (*canff* ← *Partial*) e também a quantidade disponível (*maxFreeRes*) de recursos (linha 19). Nesse último caso, os recursos são também reservados e um temporizador é iniciado para que os recursos sejam liberados quando não houver resposta a essa reserva (linha 21). No recebimento de uma mensagem *fedReqiAck*$_i$ (linha 31), o provedor de *cloud* recupera e valida o *token* de autenticação do usuário (linhas 33-34). Esta validação envolve a verificação dos dados de assinatura do *token* (data de emissão e duração do mesmo), dentre outros atributos do usuário contidos neste *token*, e corresponde ao controle de autorização de um provedor de *cloud*. A apresentação do *token* corresponde à posse de uma competência (ou *capability*) na nomenclatura de controle de acesso [Landwehr 2001].

Algoritmo 5 – *Cloud Provider*

Algoritmo 5. *Cloud C_j*

```
1.  Init:
2.  resReq ← Ø;
3.  localRes ← Ø;
4.  resPointers ← Ø;
5.  fedReqAck ←Ø;
6.  timeout ← initialValue;
7.  available ← True;
8.  δi ← Ø;

9.  upon Receive reqNotifi from RP do
10.   slau ← reqNotifi.fedReqiF.fedReqi.slai;
11.   type ← reqNotifi.type;
12.   resReq ← reqNotifi.fedReqiBi.fedReqi.res(type);
13.   idreq ← reqNotifi.idReq;
14.   if agreeSLA(slau) then
15.     localRes ← getFreeResources(resReq, type);
16.     if |localRes| ≥ | resReq| then
17.       reserveRes(idreq, localRes);
18.       canff ← Full;
19.       fedResi ← <FEDRES, idCj, idreq, canff, signcj, certcj>;
20.       send fedResi to Bi;
21.       δi.time ← time();
22.     else
23.       canff ← Partial;
24.       reserveRes(reqID, localRes);
25.       maxFreeRes ← |localRes|;
26.       fedResi ← <FEDRES, idCj, idreq, canff, ↩
              maxFreeRes, signcj, certcj>;
27.       send fedResi to Bi;
28.       δi.time ← time();
29.     end if
30.   end if

31. upon Receive fedReqiAcki from Bi
32.   idreq ←fedReqiAcki.idreq;
33.   tokenui ← fedReqjAck.authTknUi;
34.   if validadeToken(tokenui) then
35.     δi.time ←⊥;
36.     idBi ←fedReqiAcki.idBi;
37.     if (resReq = "Full") then
38.       resPointers ←Allocate(idreq);
39.     else
40.       resPointers ←Allocate(idreq [i…n]);
41.       fedAlAcki ←<FEDALCK, idCj, idBi, idreq, ↩
          resPointers, signcj, certcj>;
42.       send fedAlAckj to Bi;
43.     end if;
44.   end if;

45.   upon (time() - δi.time) ≥ timeout do
46.     freeResource(idreq);

47. upon LocaResource.Type.count ≤ 0 do
48. if available = True then
49.   memExitm ← <MEMEXIT, Cj, Type>
50.   available ← False;
51.   send memExitm to RP
52. end if

53. upon LocaResource.Type.count > 0 do
54. if available = False then
55.   memEnterm ← <MEMENTER, Cj, Type>
56.   available ← True;
57.   send memEnterm to RP;
58. end if;
```

Com a validação do *token*, o provedor remove o temporizador que controla a reserva e os recursos referentes à requisição são então alocados (linhas 35 - 40). É importante ressaltar que, no caso de atendimento parcial, o provedor de *cloud* não necessariamente aloca todos os recursos previamente reservados (*maxFreeRes,* na linha 26). O *broker*, no envio de sua mensagem $fedReq_iAck_i$, define a quantidade de recursos que deseja no atendimento parcial de cada provedor selecionado. Após alocar os recursos, o provedor de *cloud* responde ao *broker*, através da mensagem $fedAlAck_j$, indicando os ponteiros dos recursos alocados (linhas 41-42).

Os provedores devem também gerenciar a evolução dinâmica da disponibilidade de recursos. Com o decorrer do tempo, tipos de recursos podem ficar indisponíveis ou disponíveis nestes provedores. Diante destas variações, os provedores podem remover suas inscrições no *PR* para receberem notificações sobre determinados tipos de recursos, assim como podem se inscrever em listas associadas quando os recursos correspondentes estiverem novamente disponíveis. Para essas evoluções de seus registros em listas associadas de tipos de recursos, o provedor dispõe dos envios das mensagens $memEnter_m$ e $memExit_m$ para ser adicionado ou removido de listas associadas no *PR* (linhas 47 - 58).

2.2.6 Provedor de Identidades

A autenticação de usuários é importante para o acesso ao Suporte da Federação, bem como para a alocação de recursos junto aos provedores de *Cloud*. Diante disso, o modelo proposto adota um gerenciamento de identidades centralizado [Jøsang et al. 2005], onde um provedor de identidades autentica os usuários dos serviços para o *SF* e provedores de *cloud*. O algoritmo 6 mostra as interações do *IdP* da federação no modelo proposto.

Ao fazer o seu primeiro acesso ao *SF*, o usuário é direcionado ao *IdP* para se autenticar. Para isso, o *SF* envia uma requisição de autenticação ($authReq_i$) ao *IdP* da federação (linhas 8 e 9, Algoritmo 2). O *IdP*, ao receber essa mensagem (linha 3, Algoritmo 6), verifica se o usuário não está autenticado, através da existência do *token* de autenticação correspondente em sua lista de *tokens* (linha 7, Algoritmo 6). Caso o usuário já tenha sido autenticado, o *IdP* responde ao *SF* enviando o *token* recuperado (linhas 8). Caso o usuário ainda não tenha se autenticado, o *IdP* requisita então que este informe suas credencias, enviando uma mensagem $authCReq_i$ ao *Cloud User* (linhas 10 e 11). No recebimento das credenciais do usuário, através da mensagem $authCred_i$, uma vez validadas estas credencias (linhas 12-14) o *IdP* gera um *token* de autenticação devidamente assinado ($authToken_i$), o armazena e, por fim, o envia para o *SF* que solicitou a autenticação do usuário (linhas 15 - 19, Algoritmo 6).

Algoritmo 6 – Provedor de Identidades

Algoritmo 6. *Identity Provider IdP*

```
1.  Init:
2.  tokenList[];
3.  upon Receive authReq_i from F
4.    U_i ← authReq_idp.id_Ui;
5.    id_req ← authReq_Idp.id_req;
6.    authToken_i ← tokenList.get(U_i);
7.    If authToken_i ≠ null then
8.      send authToken_i to F;
9.    else
10.     authCReq_i ← <AUTHCREQ, id_IdP, id_req, Sign_IdP, Cert_IdP>;
11.     send authCReq_i to U_i;
12.   upon Receive authCred_i from U_i
13.     credentials ← authCred_ui.credentials;
14.     if verifyCredentials(credentials) then
15.       U_i ← authCred_i.id_Ui;
16.       id_req ← authCred_i.id_req;
17.       authToken_i ← <AUTHTOKEN, id_Ui, id_req, signature, ↩
                  RevokeDate, sign_idp, cert_idp>;
18.     tokenList ← tokenList ∪ authToken_i;
19.     send authToken_i to F;
20.   end if;
```

2.2.7 Considerações sobre os Algoritmos

Algumas considerações devem ser feitas sobre as operações efetuadas pelo *broker* (Algoritmo 3). Na abordagem proposta, no caso de somente um provedor atender por completo a requisição do usuário, não é verificado se este provedor é o que melhor atende a requisição (em termos de *QoS*, ou outro fator). Simplesmente o primeiro provedor de *cloud* que atende a requisição completamente é o escolhido (linhas 17-23, algoritmo 3). Um modelo que aguardasse as diversas respostas e as analisasse, usando critérios de *QoS* ou qualquer outro, poderia gerar um melhor benefício para o usuário da federação. Porém, nesse momento, este não foi o foco do trabalho. Além disso, na verificação de *SLAs* de usuário, não entramos em discussões sobre negociações dos parâmetros descritos no contrato. Na abordagem proposta o provedor de *cloud* aceita ou não o *SLA* do usuário (linha 14, Algoritmo 5). Porém um protocolo que levasse em conta uma negociação entre o que pode ser oferecido pelo provedor e as necessidades de usuário poderia tornar esta relação de uso de recursos mais flexível.

A segunda consideração a ser feita é sobre o atendimento da requisição do usuário por diversos provedores de *cloud* (linhas 24 - 33, algoritmo 3). Nesse caso, por simplicidade, foi adotada uma heurística que ordena a lista de provedores que devem atender a requisição do usuário, priorizando aqueles com maior número de recursos disponíveis. Podem ocorrer situações onde são excluídos provedores que podem atender totalmente a requisição. É o caso quando a soma dos recursos parciais é atingida antes do recebimento de uma resposta com a possibilidade citada. Assim, a primeira consideração sobre o *broker*, ou seja, a aplicação de uma heurística mais complexa que analisasse os diversos parâmetros das respostas parciais, poderia trazer benefícios. Outro ponto a ser considerado é a reserva de recursos no provedor de *cloud* (linhas 17 - 28, algoritmo 5). No momento da reserva, o provedor deixa de oferecer esses recursos a outras demandas,

o que não é interessante para os provedores. Nesse sentido, a definição do parâmetro $\delta_i.time$ (linha 28, algoritmo 5) deve levar em conta um *tradeoff* entre poder contribuir com o atendimento aos contratos e não perder outras demandas.

3. Simulações e Análises

O objetivo central que colocamos nas simulações realizadas é obter medidas aproximadas dos custos em mensagens e do tempo necessário para alocar recursos na federação. Para tanto, um modelo de simulação foi implementado. O modelo criado teve como base o simulador *CloudSim* [Calheiros et al. 2011], no qual cada entidade do modelo proposto foi mapeada como uma *CloudSim Entity*, representando os diferentes atores da federação, e as trocas de mensagens foram mapeadas como *CloudSim Events*.

3.1 Definição dos parâmetros do modelo

Vários parâmetros tiveram que ser definidos para que fosse realizada a simulação. Para cada um dos eventos trocados entre as entidades foi atribuído um valor de *delay*, representando os tempos nas trocas de mensagens. A obtenção destes valores de *delay* tomou como base experimentos realizados com a plataforma de *cloud Azure* da Microsoft. Várias medições de largura de banda e latência entre servidores de diferentes regiões geográficas foram realizadas. Utilizamos instâncias de servidores das regiões: América do Sul (BRZ), Europa (EUR), Estados Unidos (USA) e Sudeste da Ásia (ASI).

Outro parâmetro usado em nossas simulações é a quantidade de recursos (*VMs*) requeridos pelos usuários, que denominamos de *workload*. Para tanto, duas abordagens foram definidas: a primeira com o *workload* crescendo de forma linear e a outra com o *workload* na forma de uma distribuição aleatória. O *workload* com crescimento linear é constituído de provedores com um número fixo de recursos livres. Esse *workload* apresenta um aumento gradativo de pedido de recursos por teste, caracterizando um crescimento linear das demandas e definindo também o aumento de provedores atuando sobre as requisições de usuários. No outro modelo de *workload*, os recursos livres, embora fixos no sistema, são distribuídos de forma aleatória entre os diversos provedores de *cloud* da federação. O atendimento das requisições de usuários vai depender da instância de teste. Uma demanda pode ser resolvida com poucos provedores de *cloud*, enquanto em outra instância essa mesma demanda pode envolver um número muito maior de provedores.

Um parâmetro também importante para a avaliar o tempo de resposta aos usuários é o tempo aproximado para a alocação destes dos recursos em um provedor de *cloud*. Usamos novamente a plataforma *Azure* para determinação do tempo para alocar *VMs* do tipo mais básico, e as medições realizadas apontaram para um valor de cerca de 2 minutos. Devido à diferença na ordem de grandeza desse tempo se comparado aos tempos das trocas de mensagens (milissegundos), não incluímos este valor nos gráficos dos resultados. Desta forma, os gráficos dos resultados apresentam os tempos resultantes das trocas do protocolo sem o tempo de alocação dos recursos.

Os algoritmos do modelo proposto apresentam trocas de autenticação. Porém, devido à dificuldade de representar a interação entre usuário e *IdP* no processo de *login*, o modelo simulado considera que o usuário já efetuou o seu *login* na federação. Os pedidos de autenticação enviados ao *IdP* pelo *SF* ($authReq_i$ nas linhas 8 e 9 do Algoritmo 2) fazem parte das simulações. A resposta a esses pedidos é sempre o envio pelo *IdP* do *token* de autenticação ($authToken_i$, na linha 10 do algoritmo 2), considerando sempre que o usuário

já foi previamente autenticado. Por fim, definimos diferentes perfis de simulações que descrevem o posicionamento (geográfico) de cada uma das entidades da simulação. A Tabela 1 apresenta alguns dos perfis de simulação avaliados.

Tabela 1 – Perfis de Simulações

Perf.	User	SF	PR	*IdP*	CP1	CP 2	CP 3	CP 4	CP 5	CP 6	CP 7	CP 8	CP 9	CP 10
1	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ	BRZ
2	BRZ	USA	EUR	BRZ	BRZ	USA	EUR	ASI	USA	EUR	BRZ	ASI	BRZ	BRZ
3	USA	ERU	ASI	USA	BRZ	USA	EUR	ASI	BRZ	USA	EUR	ASI	BRZ	USA

3.2 Modelo de *Workload* Linear

Neste cenário, o parâmetro *workload* foi definido em forma crescente com a inclusão de um provedor a cada instância de teste, compondo uma faixa de recursos que vai de 10 até 100 *VMs*, assumindo que cada provedor fornece o valor fixo de 10 *VMs*. A Figura 2(A) apresenta resultados de tempo de resposta para o usuário obter recursos na federação (recebendo os ponteiros de recursos) em cada um dos *workloads* lineares executados nos diferentes perfis. Já a Figura 2(B) apresenta o crescimento do número de mensagens quando do aumento na alocação de recursos através da inclusão de novos provedores no atendimento dos *workloads*. É possível perceber que o número de mensagens cresce de forma linear. Quando um novo provedor é adicionado para atender a um *workload*, seis (6) novas mensagens são adicionadas aos custos de alocação.

Para prover uma visão dos resultados obtidos no workload linear, tomou-se como exemplo um usuário solicitando 100 recursos na federação e considerando o perfil 3, que dá uma maior diversidade de distribuição geográfica do modelo. Nessa situação o tempo para alocação de recursos na federação ficou em aproximadamente em 2 minutos acrescidos de 351 *ms* consumidos em trocas de comunicações. Ou seja, se tomarmos como referência os 2 minutos que são consumidos na plataforma Azure para alocar VMs, podemos dizer que os custos de alocação em uma federação atendendo um workload linear são praticamente os mesmos e que os custos de comunicação não têm um peso significativo nessas alocações. É claro que consideramos que a disponibilidade de provedores em uma federação torna as respostas imediatas a demandas de usuários.

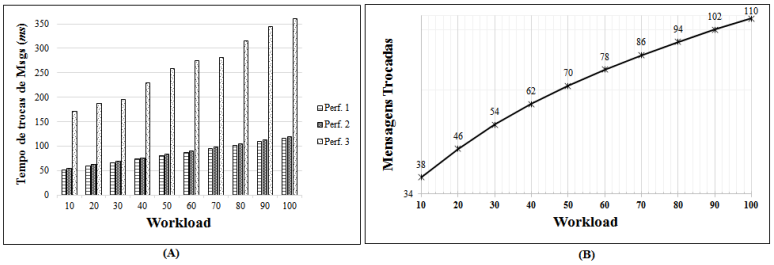


Figura 2 – Resultados do *Workload* Linear

3.3 Modelo com Workload Aleatório

Nestes testes, o *workload* do usuário foi baseado em uma distribuição normal com média em 100 e desvio padrão de 20 nos recursos solicitados. A escolha desses valores não se baseou em nenhuma observação específica, mas sim na ideia de que o número de recursos requisitados pelos usuários estaria entre 80 e 120 *VMs*, o que para grandes usuários de *cloud* pode ser considerado como valor razoável. Os recursos livres por provedor são definidos entre 5 e 150 *VMs*. A definição desses valores para os parâmetros citados foi baseada nos inúmeros testes realizados previamente. Foi observado que escolhas de faixas

menores resultavam em *workloads* sem solução (insuficiência de recursos), o que não é interessante para avaliação do modelo.

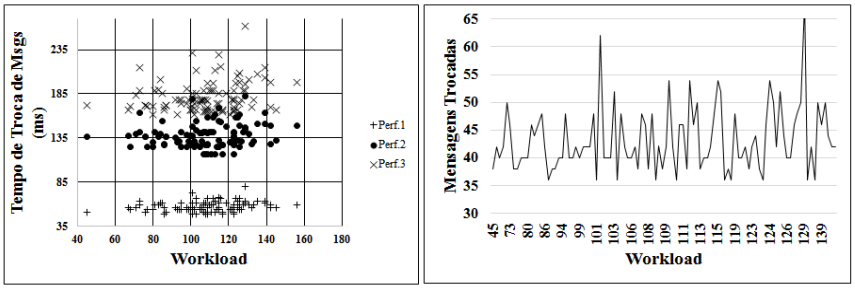


Figura 3 - Resultados do *Workload* Aleatório

As Figuras 3 (A) e (B) apresentam os resultados de tempo de trocas e do número de mensagens trocadas, considerando *workloads* aleatórios. As trocas em cada simulação pode apresentar resultados distintos e, ainda assim, serem considerados como válidos, pois uma execução otimista pode envolver o menor número possível de provedores, enquanto uma execução pessimista pode envolver o máximo de provedores disponíveis.

Tomando a mesma situação da seção anterior – ou seja, um usuário que solicita 100 recursos na federação com Perfil 3 – e considerando o *workload* aleatório, o tempo de alocação de recursos também ficou próximo de 2 minutos, mas foi acrescido de 167 *ms* devido às trocas de mensagens. É possível perceber que a simulação com *workload* aleatório teve tempo de resposta levemente menor do que na carga linear. Essa diferença se dá pelo fato de que, na simulação linear, para a requisição de 100 recursos, obrigatoriamente o suporte irá se utilizar de todos os provedores da federação. Com *workload* aleatório, a aquisição de 100 recursos se dá (na média) com um número menor de provedores.

4. Literatura Relacionada

Um dos primeiros trabalhos para federações de *clouds* foi apresentado por *Buyya* [Buyya et al. 2010], onde uma federação de *clouds* é tida como um modelo de múltiplos provedores interagindo na busca de recursos. Essa busca é em nível provedor-provedor, não havendo um suporte global para tal. Os usuários se associam a um provedor, que fica responsável por atender toda a demanda do cliente. Abordagens similares a essa são também encontradas em [Rochwerger et al. 2009],[Celesti et al. 2010].

Outras experiências de federação, como as apresentadas em [Carlini et al. 2012; Coppola et al. 2012; Villegas et al. 2012], são próximas da proposta neste artigo e fazem o uso de um suporte global na busca recursos. Estes modelos, com uma entidade central controlando buscas e alocações, fazem com que usuários de *clouds* tenham também que interagir com o suporte global que caracteriza a federação. Porém, apesar de não serem claramente discutidos, os registros de usuários podem estar ligados a domínios locais de seus provedores de *cloud* ou controlados a partir deste suporte global da federação. Na verdade, não existe clareza sobre como se dá o gerenciamento de identidades na maioria das abordagens citadas. Sendo sempre passível de interpretação se cada provedor de *cloud*, ou mesmo um suporte global, é responsável pelo seu domínio de usuários. Outra incógnita é como são executados os controles de autenticação e autorização. Estas propostas, em sua maioria, são fundamentadas em modelo tradicional de gerenciamento de

identidades, gerando certas implicações quando federações de *clouds* são definidas a partir de controles locais de identidades. Com domínios de nomes locais e o provimento de recursos se dando no nível provedor-provedor, fica difícil tarifar o usuário pelo seu uso de recursos quando a relação entre provedores pode envolver múltiplos usuários cujos identificadores só tem sentido local.

Uma exceção é a abordagem descrita em [Celesti et al. 2010], onde os autores propõem uma federação de *clouds* baseada em relações provedor a provedor. Essa proposta considera o gerenciamento de identidades federadas. Portanto, as autenticações dos usuários em suas *clouds* locais podem ser transpostas de forma a utilizar recursos de outras *clouds* na federação, sem a necessidade da reapresentação de credenciais. Porém, os custos de manutenção de base de dados com atributos de usuários são locais.

Outra distinção entre as abordagens da literatura é quanto à busca de recursos na federação de *clouds*. Duas abordagens podem ser identificadas: a primeira fazendo uso de um repositório central e a outra baseada em buscas P2P. Nas buscas através de um repositório central [Buyya et al. 2010], os provedores de *cloud* devem periodicamente informar quais são seus recursos disponíveis para a federação a partir deste mecanismo centralizador. Sempre que precisam de recursos, usuários devem recorrer ao repositório central. Apesar de facilitar a busca de recursos, essa técnica pode apresentar fragilidades: o repositório nem sempre apresenta uma visão consistente dos recursos disponíveis nos provedores do sistema distribuído de larga escala que caracteriza uma federação. Isto é consequência das características desses grandes sistemas, normalmente dispersos em complexas distribuições geográficas e sujeitos a grandes demandas.

Nos modelos que fazem uso e trocas P2P na busca de recursos [Celesti et al. 2010; Coppola et al. 2012] a ideia é que, sempre quando necessário, provedores de *cloud* trocam mensagens no sentido de verificar os recursos disponíveis na federação. Essa abordagem resolve o problema de consistência das informações existente na abordagem de repositório central, porém os custos de buscas P2P são geralmente altos em número de mensagens. Tais trocas de mensagens não garantem que os recursos estarão ainda disponíveis quando estes forem definitivamente solicitados para a alocação.

O modelo descrito no presente artigo propõe a adoção de uma estrutura diferente, baseada em um Painel de Recursos. Através do *PR*, um *broker* apresenta uma requisição em nome de seu usuário e, provedores de *cloud* são então notificados conforme seus registros como fornecedores dos tipos de recursos solicitados. Cada provedor decide se pode ou não atender tal demanda, respondendo diretamente ao *broker*, assim como reservando os recursos por um tempo determinado. O uso dessa técnica permite que o número de mensagens trocadas seja reduzido e garante que os recursos sejam alocados em determinado prazo, devido às reservas feitas.

5. Conclusão

Neste trabalho foi apresentada uma abordagem para a federação de provedores de *cloud* na qual foram descritos um modelo e os algoritmos de interação entre as entidades envolvidas. Adotamos um modelo de federação que faz uso de uma entidade global responsável pela busca e alocação de recursos em nome dos usuários na federação.

Novas abordagens foram propostas neste modelo de federação. Primeiramente, quanto à busca de recursos na federação, as demandas são postas em forma de leilão em

um Painel de Recursos. Os provedores somente respondem às demandas se podem atendê-las. Em caso contrário, não se manifestam nas negociações. Essa forma de interação torna mais simples a busca de recursos se compararmos com situações como a de repositórios globais ou serviços de informações, que dependem de atualizações frequentes feitas pelos provedores de *cloud*. Outro diferencial da abordagem proposta é quanto à inclusão de mecanismos de segurança no modelo de federação. Dos trabalhos apresentados na literatura, somente em [Celesti et al. 2010] são apresentados conceitos de autenticação e autorização para acesso a recursos na federação, porém sem a definição dos protocolos necessários e suas implicações. O modelo proposto emprega controles de segurança baseados no modelo de gerenciamento de identidades centralizado, que foi adotado por ser mais simples e adequado à proposta de federação descrita neste artigo.

Agradecimentos

Agradecemos a CAPES pelo apoio financeiro com bolsa de doutorado e bolsa CNPQ Processo [233648/2014-3].

Referências

Buyya, R., Ranjan, R. and Calheiros, R. (2010). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. International Conference on Algorithms and Architectures for Parallel Processing, p. 13–31.

Calheiros, R. N., Ranjan, R., Beloglazov, A. and Rose, A. F. De (2011). CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. n. August 2010, p. 23–50.

Carlini, E., Coppola, M., Dazzi, P., Ricci, L. and Righetti, G. (2012). Cloud Federations in Contrail. Euro-Par 2011 Workshops, p. 159–168.

Celesti, A., Tusa, F., Villari, M. and Puliafito, A. (2010). How to Enhance Cloud Architectures to Enable Cross-Federation. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on.

Coppola, M., Dazzi, P., Lazouski, A., et al. (2012). The Contrail approach to cloud federations. The Internation Symposium on Grids and Clouds (ISGC) 2012,

Grozev, N. and Buyya, R. (2012). Inter-Cloud architectures and application brokering: taxonomy and survey. Software: Practice and Experience, p. 1–22.

Jøsang, A., Fabre, J., Hay, B., Dalziel, J. and Pope, S. (2005). Trust requirements in identity management. CRPIT '44: Proceedings of the 2005 Australasian workshop on Grid computing and e-research, p. 99–108.

Landwehr, C. E. (2001). Computer security. International Journal of Information Security, v. 1, n. July, p. 3–13.

Miller, J. (2006). Yadis Specification 1.0.

Rochwerger, B., Breitgand, D., Levy, E., et al. (jul 2009). The Reservoir model and architecture for open federated cloud computing. IBM Journal of Research and Development, v. 53, n. 4, p. 4:1–4:11.

Villegas, D., Bobroff, N., Rodero, I., et al. (sep 2012). Cloud federation in a layered service model. Journal of Computer and System Sciences, v. 78, n. 5, p. 1330–1344.

Seleção de Métricas Efetivas na Detecção de Anomalias em Sistemas Multi-camadas usando Correlação Parcial

Otto Julio Ahlert Pinno[1,2], Sand Luz Correa[1], Aldri Luiz dos Santos[2], Kleber Vieira Cardoso[1]

[1]Instituto de Informática (INF) – Universidade Federal de Goiás (UFG)

[2]Departamento de Informática – Universidade Federal do Paraná (UFPR)

{ottosilva,sand}@inf.ufg.br, aldri@inf.ufpr.br, kleber@inf.ufg.br

Abstract. *Large-scale data centers allow organizations to gain access to computer resources without incurring high costs in purchasing and maintaining IT infrastructure. In these environments, due to the large number of hardware and software involved, anomaly detection is difficult but essential for service provisioning. Computer systems hosted in data centers usually involve multiple layers and provide a large set of metrics for tracking their operation. The analysis of all available metrics generates drawbacks associated with communication, storage and processing. A more efficient way to support anomaly detection and minimize the cost of monitoring is to use stable statistical correlations among metrics that reflect the system state. We present the strategies PCTN, MST-PCTN and PCTN-MST, based on partial correlation, for selecting metrics to support anomaly detection in multi-tier systems. We evaluate the proposed strategies using an e-commerce, Web transaction benchmark. Results show that the PCTN-MST strategy allowed the construction of a monitoring network with 8% less metrics than that obtained with MST and achieved a fault coverage up to 10% larger.*

Resumo. *Centros de dados de larga escala permitem que organizações tenham acesso a recursos computacionais sem incorrer em alto custo de aquisição e manutenção de infraestrutura de TI. Nesses ambientes, devido aos inúmeros recursos de hardware e software envolvidos, a detecção de anomalias, embora difícil, se torna essencial à manutenção dos serviços aos usuários. Sistemas computacionais hospedados em centros de dados geralmente envolvem múltiplas camadas e disponibilizam um grande conjunto de métricas para coleta de dados sobre o seu funcionamento. A análise de todas as métricas disponíveis gera inconvenientes associados à comunicação, armazenamento e processamento dos dados. Uma forma mais eficiente de apoiar a detecção de anomalias e minimizar o custo do monitoramento é o emprego de correlações estatísticas estáveis entre métricas que refletem o estado do sistema. Neste trabalho, apresentamos as estratégias PCTN, MST-PCTN e PCTN-MST, baseadas em correlação parcial, para a seleção de métricas que apoiam a detecção de anomalias em sistemas multi-camadas. Avaliamos essas estratégias usando um benchmark de transações Web. Os resultados mostram que a estratégia PCTN-MST permitiu a construção de uma rede de monitoramento com* 8% *a menos de métricas que a estratégia MST e alcançou uma cobertura de falhas até* 10% *maior.*

1. Introdução

Cada vez mais organizações têm feito uso de infraestruturas de computação providas por centros de dados de larga escala. Esse modelo oferece flexibilidade pois permite que empresas tenham acesso a recursos sob demanda, sem a necessidade de incorrer em alto custo de aquisição e manutenção de infraestruturas de TI. No entanto, a operação e o gerenciamento de um centro de dados é uma tarefa complexa devido ao grande número de recursos de hardware e software que geralmente constituem esse ambiente [Aceto et al. 2013].

Tipicamente, centros de dados hospedam aplicações *Web*, tais como serviços de comércio eletrônico, redes sociais e serviços de transmissão de conteúdo multimídia. Essas aplicações são, em sua grande maioria, sistemas complexos compostos por várias camadas, como a de apresentação provida por um servidor HTTP (por exemplo o Apache), a de lógica da aplicação provida por um servidor de negócio (por exemplo o Tomcat) e a de persistência provida por um servidor de banco de dados (por exemplo o MySQL). Cada camada é um serviço independente e a sobreposição de eventos como *locks* de dados, variabilidade de tempo de serviço de uma operação, contenção de memória e concorrência, nas diferentes camadas, pode levar a padrões de execução anormais ou até mesmo a interrupção completa do serviço oferecido para o usuário [Mi et al. 2010].

Apesar dessas dificuldades, é esperado que as aplicações *Web* apresentem alta disponibilidade, confiabilidade e responsividade, sob pena de causarem perdas econômicas expressivas [Wang et al. 2013]. Estima-se que a parada ou interrupção de *Web sites* empresariais tem um custo médio de US$ 21 mil por hora para suas organizações proprietárias [Simic 2014]. O alto custo de parada dos serviços *online* atuais aliado à complexidade de grandes aplicações *Web* levam à necessidade de ferramentas que possam auxiliar os administradores a rastrear o comportamento desses sistemas e detectar padrões de execução anormais. Para auxiliar o desenvolvimento dessas ferramentas, sistemas de computação que operam em centros de dados disponibilizam uma grande quantidade de dados sobre o funcionamento de diversos componentes, incluindo dados de arquivos de *log* dos componentes das aplicações, métricas relacionadas às tecnologias de *middleware* usadas nos sistemas, eventos de auditoria, além de estatísticas relacionadas a métricas do sistema operacional e tráfego da rede. No entanto, a coleta de todos esses dados afeta não apenas o desempenho dos sistemas mas também gera inconvenientes associados à comunicação, armazenamento e processamento dos dados.

Uma abordagem mais eficiente para apoiar a detecção de anomalias em sistemas que seguem uma arquitetura multi-camadas composta por vários serviços, como as aplicações *Web*, consiste no uso de correlações estatísticas estáveis entre métricas que refletem o estado do sistema [Jiang et al. 2006, Jiang et al. 2009, Munawar et al. 2009]. Nessa abordagem, correlações entre métricas são capturadas através de modelos matemáticos que descrevem o comportamento de uma métrica em função do estado de outra. Como apenas correlações estáveis são consideradas, é esperado que as relações descritas pelos modelos mantenham-se válidas enquanto o sistema operar livre de falhas. No entanto, essas relações serão violadas quando ocorrerem falhas no sistema. A seleção de correlações estáveis tem também o efeito de filtrar um subconjunto de métricas dentre o conjunto disponível. Dessa forma, uma vez encontradas as correlações estáveis, apenas as métricas envolvidas nessas correlações passam a ser coletadas e monitoradas periodicamente, reduzindo o custo de monitoramento. Uma dificuldade, no entanto,

consiste em encontrar tais correlações. Existem algumas soluções para esse problema, tais como a validação de modelos de forma iterativa [Jiang et al. 2006] ou a seleção de correlações estáveis usando coeficiente de Pearson [Magalhaes and Silva 2010]. Porém essas soluções apresentam restrições, discutidas na Seção 2.

Neste trabalho, investigamos uma nova abordagem baseada em correlação parcial [Baba et al. 2004], a qual descreve o quanto o relacionamento entre duas variáveis (métricas) é resultado de suas correlações com uma variável intermediária. Essa informação permite excluir relacionamentos que são resultados de correlações indiretas, obtendo um conjunto mínimo de correlações estáveis. Neste trabalho, apresentamos as estratégias PCTN, MST-PCTN e PCTN-MST, baseadas em correlação parcial, para a seleção de métricas que apoiam a detecção de anomalias em sistemas multi-camadas. Avaliamos a efetividade dessas estratégias através de um conjunto de experimentos detalhados usando o TPC-W [Menasce 2002], um *benchmark* de transações *Web* para sistemas de comércio eletrônico. Os resultados mostram que a estratégia PCTN-MST permitiu a construção de uma rede de monitoramento com 8% a menos de métricas que aquela obtida usando MST, a melhor estratégia existente atualmente para esse problema, e alcançou uma cobertura de falhas até 10% maior.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta alguns fundamentos e trabalhos relacionados. A Seção 3 descreve as estratégias propostas. A Seção 4 apresenta a avaliação experimental e os resultados encontrados. Finalmente, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Fundamentos e Trabalhos Relacionados

Ferramentas comerciais como o SmartCloud [IBM 2014] e o System Center Operations Manager [Microsoft 2013] frequentemente adotam soluções baseadas em regras para detectar anomalias em centros de dados. Nessas soluções, os administradores configuram manualmente regras que estabelecem limiares para algumas métricas a serem monitoradas. Quando o valor de uma métrica extrapola seu limiar, uma anomalia é detectada e um alarme é emitido para o administrador. No entanto, a observação individualizada de métricas não é suficiente para descrever o comportamento de um sistema complexo [Jiang et al. 2006, Chen et al. 2010, Wang et al. 2014].

Uma forma mais eficaz consiste em observar também possíveis correlações entre métricas, pois essas correlações podem capturar a dinâmica do sistema à medida que as requisições de usuários fluem através dos componentes da aplicação. Considere, por exemplo, uma aplicação *Web*. Se existe uma correlação direta entre o número de requisições HTTP para o servidor *Web* e o número de requisições SQL para o servidor de banco de dados, então podemos esperar que um aumento de requisições no primeiro servidor leve a um aumento de requisições no segundo. Se esse relacionamento é estável, ou seja, ele se mantém durante a operação normal do sistema, podemos tomá-lo como uma invariante. Quando uma falha ocorre, a dinâmica do sistema é afetada e algumas invariantes são violadas. Portanto, podemos detectar falhas num sistema monitorando suas invariantes, ou seja, suas correlações estáveis. Esta forma de descrever o comportamento de um sistema é denominado *monitoramento baseado em correlação* [Jiang et al. 2009].

O monitoramento baseado em correlação apresenta diversas vantagens. Essa abordagem não exige nenhuma informação de entrada por parte do administrador, pois ne-

nhum conhecimento prévio da estrutura do sistema é requerido. Além disso, essa abordagem é genérica, podendo ser aplicada em qualquer sistema com monitoramento de dados. Por fim, a seleção de correlações estáveis reduz o número de métricas monitoradas, reduzindo a sobrecarga imposta pelo monitoramento. No entanto, o desafio nessa abordagem é encontrar as correlações verdadeiramente estáveis.

Para resolver esse problema, os trabalhos existentes em monitoramento baseado em correlação utilizam diferentes abordagens. [Jiang et al. 2006] usam regressão linear para capturar o relacionamento entre pares de métricas monitoradas por um sistema *Web*. Para determinar quais relações lineares são estáveis, uma vez que os modelos são gerados para todas as combinações possíveis envolvendo duas métricas, os autores usam um método iterativo e estabelecem um fator de confiança, o qual é medido periodicamente a partir do número de previsões corretas emitidas pelos modelos. Quando o fator de confiança de um modelo cai abaixo de um limiar, ele é removido da solução. Um problema dessa abordagem é sua convergência lenta para o conjunto mínimo de métricas.

Uma alternativa ao método iterativo proposto por [Jiang et al. 2006] consiste em calcular o coeficiente de Pearson para quantificar a intensidade do relacionamento entre duas métricas. Essa abordagem foi utilizada por [Magalhaes and Silva 2010] para detectar problemas de desempenho em aplicações *Web*. Dadas n observações de duas variáveis X e Y, o coeficiente de Pearson entre essas variáveis é denotado pela Equação 1 e representa o quanto as duas variáveis mudam de maneira similar (covariância) relativamente a suas dispersões (desvios padrões):

$$\rho(X,Y) = \frac{cov(X,Y)}{s_X s_Y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}, \quad (1)$$

onde X_i e Y_i são observações das variáveis X e Y, respectivamente, e $\bar{X}$ e $\bar{Y}$ são as médias das n observações de X e Y, respectivamente. ρ assume valores no intervalo $[-1, 1]$. Em geral, assume-se que a correlação é forte se o valor absoluto de ρ está no intervalo $[0.5, 1]$ [Cohen 1988]. A definição de correlação forte pode ser usada para encontrar as correlações estáveis. Nesse sentido, consideramos estáveis apenas as correlações fortes.

Uma limitação do coeficiente de Pearson é que ele não provê nenhuma informação sobre a existência de uma terceira métrica condicionando o relacionamento observado entre duas outras métricas. Como consequência, o coeficiente de Pearson pode retornar um grande número de correlações indiretas. Sejam X, Y e Z três métricas onde identificamos uma forte correlação entre os pares (X, Y), (X, Z) e (Y, Z). Dizemos que a correlação entre o par (X, Y) é indireta, se ela é resultado das correlações individuais de X e Y com Z. Correlações indiretas representam informações redundantes e, portanto, podem ser eliminadas do monitoramento.

[Munawar et al. 2009] apresentam uma abordagem baseada em árvore geradora mínima (*Minimum Spanning Tree* - MST) para eliminar correlações indiretas. Após modelar as relações entre todos os pares possíveis de métricas, os autores utilizam o coeficiente de Pearson para selecionar as correlações estáveis. As correlações selecionadas são usadas para construir uma rede onde métricas são mapeadas em vértices e correlações em arestas. As correlações mais importantes dessa rede são então capturadas usando a MST.

Diferentemente de [Munawar et al. 2009], neste trabalho, usamos o conceito de *correlação parcial* para eliminar as correlações indiretas e encontrar um conjunto mínimo de correlações estáveis que garanta cobertura de falhas. A correlação parcial permite determinar se o relacionamento entre um par de métricas (X, Y) é condicionado por outra métrica Z. Essa verificação é realizada removendo a influência de Z e recalculando a correlação entre o par (X, Y). Se o resultado da correlação parcial for significativamente menor que o da correlação original, então a correlação original existia, em sua maior parte, devido à correlação de X e Y com Z. A correlação parcial é expressa pelo coeficiente de correlação parcial. Dadas duas variáveis X e Y e uma variável condicionante Z, o coeficiente de correlação parcial pode ser expresso em termos dos coeficientes de Pearson $\rho(X, Y)$, $\rho(X, Z)$ e $\rho(Y, Z)$ como mostrado na Equação 2:

$$\rho(X, Y : Z) = \frac{\rho(X, Y) - \rho(X, Z)\rho(Y, Z)}{\sqrt{[1 - \rho^2(X, Z)][1 - \rho^2(Y, Z)]}}. \quad (2)$$

A correlação parcial tem sido empregada em áreas como Biologia e Economia para estudar o comportamento de sistemas complexos [De La Fuente et al. 2004, Kenett et al. 2010]. Nesses trabalhos, a correlação parcial é aplicada para eliminar muitas correlações que não representam relacionamentos reais. Em redes ou sistemas complexos, isso é útil pois reduz o conjunto de correlações a ser analisado. Até onde sabemos, este é o primeiro trabalho que usa correlação parcial para estudar redes de métricas monitoradas em um sistema computacional.

Finalmente, é importante mencionar que alguns trabalhos utilizam busca baseada em projeção estatística para selecionar um conjunto de métricas para o monitoramento, dado um critério de interesse. PCA (*Principal Component Analysis*) é uma dessas abordagens, onde o critério é a variância das amostras. No entanto, essa técnica não é aplicável para sistemas multi-camadas, pois a carga de trabalho nesses sistemas é variável e nem sempre apresenta um comportamento regular [Peiris et al. 2014].

3. Monitoramento Baseado em Correlação Usando Correlação Parcial

Antes de introduzirmos as estratégias baseadas em correlação parcial que propusemos para selecionar correlações estáveis, descrevemos, na Subseção 3.1, o arcabouço que adotamos para detecção de anomalias em sistemas multi-camadas no contexto de monitoramento baseado em correlação. As estratégias são apresentadas na Subseção 3.2.

3.1. Modelagem do Comportamento do Sistema e Detecção de Anomalias

Usando dados de métricas coletadas de um sistema multi-camadas operando livre de falhas, dividimos esses dados em dois conjuntos: um de treinamento e outro de teste. Em seguida, implementamos as 4 atividades descritas a seguir e ilustradas na Figura 1.

1. **Identificação de correlações estáveis**: para cada par de métricas X e Y monitoradas, consideramos que existe uma correlação estável entre X e Y se, para qualquer métrica Z diferente de X e de Y, a correlação parcial $\rho(X, Y : Z)$ é maior que um certo limiar. A identificação de correlações estáveis é feita sobre o conjunto de treinamento.

2. **Construção de modelos**: para cada par de métricas X e Y para o qual o passo anterior identificou a existência de uma correlação estável, utilizamos uma técnica de regressão

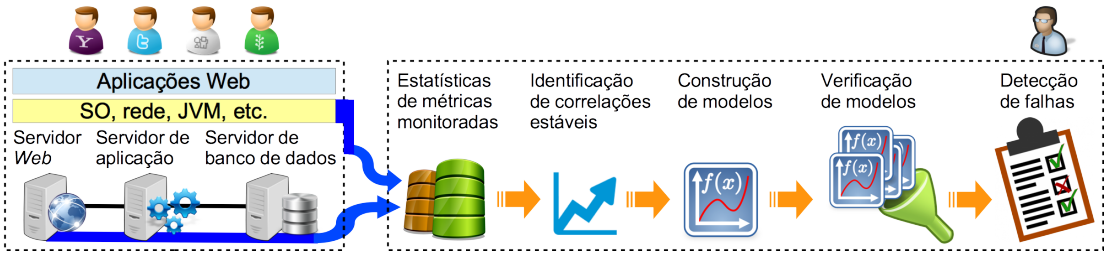


Figura 1. Visão geral do mecanismo de monitoramento.

linear sobre o conjunto de treinamento e construímos um modelo matemático $Y = f(X)$ para descrever o relacionamento entre esse par de métricas.

3. **Verificação de modelos**: para cada modelo matemático construído no passo anterior, verificamos mais uma vez a sua estabilidade, usando o conjunto de teste. Para realizar essa verificação, para cada modelo, construímos um *limite de aceitação de resíduo*. Sejam X_i e Y_i os i-ésimos valores observados para as métricas X e Y, respectivamente, no conjunto de teste. Seja $\hat{Y}_i$, o valor previsto pelo modelo quando X_i é fornecido como entrada. Definimos o resíduo como sendo a diferença entre o valor realmente observado para Y em um instante de tempo e o valor estimado pelo modelo, ou seja, $R_i = Y_i - \hat{Y}_i$.

Para determinar os limiares superior e inferior do limite de aceitação de um modelo, usamos a mesma abordagem empregada em [Jiang et al. 2006]. Seja R_i, $i = 1...n$, os resíduos gerados pelo modelo. Essa abordagem consiste em encontrar um valor $\hat{r}$ que é maior que $99,5\%$ dos resíduos observados e amplificar o resultado em 10%, como descrito na Equação 3. Os valores positivos e negativos do limiar resultante (τ) são usados, respectivamente, como limiares superior e inferior do limite de aceitação do modelo.

$$\tau = 1,1 \times arg_{\hat{r}}\{prob(|R_i| \leq \hat{r}) = 0.995\} \tag{3}$$

A verificação de um modelo é realizada fornecendo os valores X_i, do conjunto de teste, como entrada e observando os resíduos R_i gerados. É considerado estável o modelo em que até uma determinada porcentagem de observações de resíduos estão fora do limite de aceitação. Chamamos esse limiar de *Porcentagem de Resíduos Discrepantes Permitidos* - PRDP.

4. **Detecção de falhas**: a detecção de anomalias ocorre através do monitoramento dos modelos que passaram no teste de verificação. Periodicamente, esses modelos são alimentados com novas observações de métricas e os resíduos são calculados. Se uma porcentagem de resíduos maior que PRDP viola o limite de aceitação, o modelo correspondente é violado, indicando a ocorrência de uma anomalia ou falha.

É importante mencionar que as métricas monitoradas ficam restritas àquelas usadas pelos modelos que passaram no teste de verificação, reduzindo o custo de monitoramento. Esses modelos foram selecionados a partir da identificação das correlações estáveis. Essa seleção prévia acelera a convergência para um conjunto pequeno de modelos e, consequentemente, de métricas monitoradas. A seguir, descrevemos três estratégias que propusemos para identificar correlações estáveis em um sistema multi-camadas.

3.2. Estratégias para Seleção de Correlações Estáveis

Na Equação 2, um valor alto de $\rho(X, Y : Z)$ indica que Z exerce pouca ou nenhuma influência na correlação entre o par de métricas (X, Y). Isso ocorre porque o coeficiente de correlação parcial assume um valor alto apenas se $\rho(X, Y)$ for muito maior que $\rho(X, Z).\rho(Y, Z)$. Por outro lado, um valor pequeno de $\rho(X, Y : Z)$ pode indicar duas situações: (1) Z tem uma forte influência na correlação entre o par (X, Y), ou seja, $\rho(X, Y) \sim \rho(X, Z).\rho(Y, Z)$, (2) a correlação entre as três métricas é pequena, ou seja, os coeficientes de Pearson $\rho(X, Y)$, $\rho(X, Z)$ e $\rho(Y, Z)$ são pequenos. Para diferenciar entre essas duas situações, frequentemente, a correlação parcial é calculada através da influência de correlação [Kenett et al. 2010], mostrada na Equação 4:

$$d(X, Y : Z) \equiv \rho(X, Y) - \rho(X, Y : Z). \tag{4}$$

A influência de correlação quantifica a influência de Z no relacionamento do par de métricas (X, Y). Esse valor é alto apenas quando uma parte significativa da correlação entre X e Y é explicada em função da métrica Z.

Nossa primeira estratégia para identificar as correlações estáveis em um sistema multi-camadas consiste em modelar uma rede de monitoramento usando apenas a influência de correlação. Essa estratégia é denominada *Partial Correlation Threshold Network* - PCTN por utilizar o algoritmo PCTN proposto por [Kenett et al. 2010].

O algoritmo PCTN cria uma rede composta pelas influências de correlação $d(X, Y : Z)$ com valores maiores que um certo limiar. Essa rede é representada por um grafo, onde os vértices são as métricas monitoradas e as arestas correspondem às correlações entre as métricas. Para cada combinação de três métricas X, Y e Z, acrescentamos uma aresta entre Z e X e outra aresta entre Z e Y, indicando a influência de Z sobre X e Y, se, e somente se, elas ainda não existem na rede e a Equação 5 for satisfeita:

$$d(X, Y : Z) \geq \langle d(X, Y : Z)\rangle_Z + k\sigma_Z(d(X, Y : Z)), \tag{5}$$

onde $\langle d(X, Y : Z)\rangle_Z$ e $\sigma_Z(d(X, Y : Z))$ são, respectivamente, a média e o desvio padrão determinados com respeito à métrica condicionante Z. O parâmetro k, denominado limiar de influência, define a intensidade de atuação do limiar.

A topologia da rede a ser construída na PCTN depende fortemente de k. Valores altos podem levar a um grafo pouco conexo, o que prejudica a capacidade de detecção de anomalias de um mecanismo. Isso ocorre porque cada aresta representa uma correlação e cada correlação dará origem a um modelo que será monitorado. Se existem poucas arestas no grafo, o mecanismo irá monitorar poucos modelos e alguns relacionamentos importantes podem não ser representados. Por outro lado, valores muito pequenos podem aproximar a topologia da rede de um grafo completo, diminuindo a capacidade de filtragem de correlações da rede. Assim como [Kenett et al. 2010], utilizamos uma abordagem empírica para a escolha de k. Entretanto, nossa abordagem é diferente da original, como descreveremos na Subseção 4.2.1.

As duas outras estratégias propostas neste trabalho são combinações das estratégias PCTN e MST. A estratégia MST-PCTN identifica as correlações estáveis da seguinte forma. Dado o conjunto contendo todas as métricas monitoradas, calculamos

a correlação (de Pearson) para todos os pares possíveis de métricas. O conjunto de correlações é reduzido, inicialmente, construindo uma rede de monitoramento usando a estratégia MST proposta por [Munawar et al. 2009]. Em seguida, aplicamos a estratégia PCTN sobre a rede resultante, gerando outra rede. As arestas da rede final representam as correlações estáveis.

A estratégia PCTN-MST segue a mesma lógica, porém as estratégias são aplicadas na ordem inversa. Dado o conjunto contendo todas as métricas monitoradas, calculamos a correlação (parcial) para cada combinação de três métricas usando a estratégia PCTN. Em seguida, aplicamos a estratégia MST sobre a rede formada pela PCTN, gerando outra rede. As arestas da rede final representam as correlações estáveis.

4. Avaliação de Desempenho

Para realizar a avaliação das estratégias propostas, preparamos um ambiente de testes com uma aplicação *Web* multi-camadas, o qual é descrito em detalhe na Subseção 4.1. Além dessas estratégias, implementamos e avaliamos outras abordagens, inclusive a MST proposta por [Munawar et al. 2009], conforme mostraremos na Subseção 4.2.

4.1. Infraestrutura de Testes e Carga de Trabalho

A Figura 2 ilustra o ambiente de testes implantado para a realização dos experimentos. Como pode ser visto, o ambiente é uma arquitetura de três camadas, formada por: 1) camada de dados, 2) camada de negócio ou lógica e 3) camada de apresentação. A camada de dados (`Database Server`) é implementada pelo servidor MySQL, enquanto a camada de negócio (`Front Server`) é baseada no servidor de aplicação Apache Tomcat. No `Front Server`, também está instalada a lógica da aplicação *Web* que se baseou no núcleo do TPC-W. O TPC-W é um *benchmark* clássico que emula comportamentos comuns a *sites* de comércio eletrônico [Menasce 2002]. Neste trabalho, utilizamos uma implementação em Java de código aberto do TPC-W [Bezenek et al. 2014]. A camada de apresentação (`Client`) também é implementada a partir do TPC-W através de um módulo que emula clientes da aplicação *Web*. Os equipamentos `Front Server` e `Database Server` também executam a ferramenta collectd [Forster 2014] para realizar a coleta de estatísticas das métricas monitoradas. A Tabela 1 resume as principais características dos equipamentos. Todos os equipamentos utilizam o sistema operacional Linux com a distribuição Ubuntu 12.04 LTS.

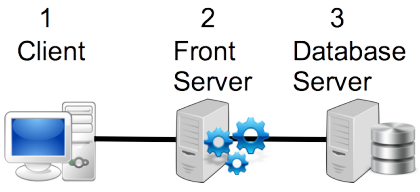


Figura 2. Arquitetura do ambiente de testes.

Equip.	Processador	Memória	Rede
1	Atom @ 1,66 GHz	1 GB	1 Gbps
2	Core i5 @ 3,20 GHz	4 GB	1 Gbps
3	Core i5 @ 3,20 GHz	4 GB	1 Gbps

Tabela 1. Especificação básica dos equipamentos.

Tipicamente, a carga de trabalho de uma aplicação *Web*, como comércio eletrônico e *sites* empresariais, pode ser descrita adequadamente no nível de sessão [Zhang et al. 2007]. Uma sessão consiste em uma sequência de requisições individuais e consecutivas. Todas as requisições individuais são necessárias para completar uma transação de mais alto nível. Portanto, uma medida de desempenho comumente

usada em aplicações *Web* é o número de sessões concorrentes que o sistema pode suportar sem violar um tempo de resposta transacional máximo estabelecido previamente.

De acordo com a especificação do TPC-W, o número de sessões concorrentes, também chamadas de navegadores emulados (*emulated browsers* – EB), é constante durante todo o experimento. Para cada *EB*, o TPC-W define, de forma estatística, o tamanho da sessão do cliente, o tempo de pensar (*think time*) do usuário e as consultas que são geradas pela sessão. O tamanho médio das sessões é de 15 minutos e sempre que uma sessão termina, uma nova é criada. Além disso, o tamanho do banco de dados é determinado pelo número de itens disponíveis para compra e também pelo número de clientes.

O TPC-W define 14 transações e 3 grupos de distribuição de acesso entre essas transações, a saber: *Browsing Mix*, *Shopping Mix* e *Ordering Mix*. A título de ilustração, a Tabela 2 lista 6 das transações e as porcentagens médias de requisições destinadas às transações de cada grupo de distribuição. Em nossa avaliação, optamos pelo *Browsing Mix* para a geração da carga, uma vez que esse grupo representa um padrão de acesso fortemente baseado em leitura. Esse tende a ser o padrão mais comum em ambientes reais, pois grande parte das transações realizadas em *sites* de comércio eletrônico envolvem pesquisas que não resultam efetivamente em pedido ou compra.

Interação Web	**Browsing Mix**	**Shopping Mix**	**Ordering Mix**
Pesquisa	**95%**	**80%**	**50%**
Home	29,00%	16,00%	9,12%
Product Detail	21,00%	17,00%	12,35%
Search Request	12,00%	20,00%	14,54%
Pedidos	**5%**	**20%**	**50%**
Shopping Cart	2,00%	11,60%	13,53%
Customer Reg.	0,82%	3,00%	12,86%
Buy Request	0,75%	2,60%	12,73%

Tabela 2. Distribuição das requisições entre as transações do TPC-W.

Inicialmente, avaliamos o sistema livre de falhas, pois o intuito era construir e treinar os modelos que descreveriam a relação entre os pares de métricas. Através de experimentos, verificamos que o sistema é capaz de tratar até 300 EBs sem apresentar atrasos sensíveis na coleta de estatísticas. Realizamos mais de 50 experimentos de 30 minutos, nos quais eram monitoradas 396 métricas. Essas métricas incluem estatísticas sobre utilização de CPU, memória, disco e rede, além de métricas do sistema operacional e da máquina virtual Java (JVM). A coleta de estatísticas é realizada ao final de cada janela de monitoramento que tem tamanho fixo de 5 segundos. Esse intervalo ofereceu um compromisso adequado entre a quantidade de amostras e o impacto no consumo de CPU, o qual se mostrou negligenciável, permanecendo abaixo de 0,5%.

4.2. Resultados

Organizamos a apresentação dos resultados em três partes, as quais são descritas nas subseções que seguem: 1) quantidade de correlações e métricas no sistema livre de falhas, 2) estabilidade das correlações em função do percentual tolerado de resíduos discrepantes e 3) desempenho em termos de percentual de falhas detectadas e taxa de falso-positivos.

4.2.1. Correlações e Métricas

A partir da avaliação do sistema livre de falhas, formamos uma base de dados de treinamento usada para criar as redes de monitoramento. Essas redes foram criadas aplicando as estratégias PCTN, MST-PCTN e PCTN-MST. Além dessas redes, criamos também outras, a partir da mesma base de dados, aplicando as estratégias de Pearson e MST. A seguir, apresentamos o processo de criação de cada rede e a quantidade de correlações e métricas resultante em cada uma. É importante mencionar que essas redes foram criadas como saída da atividade 1 descrita na Seção 3.1.

Rede de Pearson

A partir da base de dados gerada pela carga *Browsing Mix*, calculamos o coeficiente de Pearson para cada par de métricas monitoradas. Em seguida, selecionamos todos os pares que apresentaram correlação forte, ou seja $\rho > 0,5$. Do total de 78.210 correlações que abrangiam as 396 métricas, foram identificadas como fortes **1.773 correlações** que envolviam **192 métricas**. Ou seja, o monitoramento dessa rede ainda possui custo alto.

PCTN

A partir da base de dados gerada pela carga *Browsing Mix*, calculamos a influência de correlação $d(X, Y : Z)$ para cada combinação de métricas X, Y e Z, conforme a Equação 4. A seguir, construímos o grafo da rede, acrescentando uma aresta entre Z e X e entre Z e Y, em todas as combinações em que a influência de correlação satisfazia a Equação 5. Conforme descrito previamente, essa expressão exige a definição de um limiar de influência k. O valor adequado de k deve ser suficientemente alto para reduzir de maneira significativa o número de arestas, mas também deve ser baixo o bastante para manter um componente conexo com grau razoavelmente alto na rede.

[Kenett et al. 2010] adotaram uma abordagem experimental para encontrar um k adequado, o qual consiste em variar o valor desse parâmetro de forma iterativa e avaliar, para cada rede formada, a soma dos pesos das arestas da rede e o grau do vértice mais conexo. O procedimento adotado para dar pesos às arestas é: para cada aresta $Z \rightarrow X$, o peso dessa aresta é igual ao número de variáveis Y que satisfazem a Equação 5. Seguindo uma estratégia similar, variamos o valor de k e analisamos as duas informações apresentadas por [Kenett et al. 2010] sobre as redes formadas, além de verificarmos a quantidade de influências retornadas pela PCTN.

Diferentemente de [Kenett et al. 2010], não observamos uma redução abrupta no somatório dos pesos das arestas ao mesmo tempo em que o grau do vértice mais conexo se mantinha relativamente estável. Em seu estudo, [Kenett et al. 2010] mostraram que algumas ações do mercado financeiro possuem alta influência sobre uma grande parte de outras ações. No entanto, no sistema computacional investigado, observamos que as influências tendem a ser mais distribuídas entre suas métricas. Por essa razão, tivemos que adequar essa metodologia para escolha de k e o fizemos da maneira descrita a seguir. Variamos iterativamente o valor desse parâmetro até que o número máximo de correlações retornadas pela PCTN fosse igual ao número de correlações existentes na rede de Pearson. Como duas correlações podem ser originadas de cada influência retornada pela Equação 5, a saber, uma correlacionando Z e X e a outra Z e Y, o número de influência retornada deve ser igual à metade do número de correlações na rede de Pearson. Para a carga

analisada, o número de influências definido foi alcançado com $k = 17,5$. Usando esse valor como limiar de influência na Equação 5, obtivemos uma rede com **396 correlações** e **177 métricas**. Logo, a PCTN apresenta uma demanda de monitoramento sensivelmente menor que a estratégia baseada em Pearson.

Rede MST

Conforme proposto por [Munawar et al. 2009], uma MST é construída para selecionar as métricas mais importantes em uma rede de Pearson. Dado um conjunto de métricas monitoradas, é inicialmente construída a rede de Pearson para selecionar as correlações fortes. Em seguida, cada aresta da rede é rotulada com um peso igual a $1 - \rho(X, Y)$, onde $\rho(X, Y)$ é o valor do coeficiente de Pearson da correlação representada pela aresta. Por fim, a MST é calculada para a rede rotulada. As arestas da MST representam as correlações identificadas como estáveis. A MST obtida possui **183 correlações** envolvendo **192 métricas**, isto é, a rede MST possui menos correlações que a PCTN, embora envolva um número maior de métricas.

MST-PCTN e PCTN-MST

Além de avaliar o desempenho da PCTN em relação à MST, analisamos o comportamento de redes formadas pela combinação das duas, a saber: MST-PCTN e PCTN-MST. A MST-PCTN é formada aplicando o filtro baseado em correlação parcial sobre as métricas selecionadas pela MST. Para tanto, definimos $k = 15,13$ de maneira que a PCTN retornasse uma quantidade de influências de correlação igual à metade da quantidade existente na MST, ou seja, a mesma estratégia usada com a rede Pearson. Em termos de custo de monitoramento, essa é a melhor abordagem, pois obtemos uma rede com apenas **62 correlações** e **42 métricas**. No entanto, conforme mostraremos posteriormente, essa rede apresentou baixa capacidade de detecção de falhas.

Inicialmente, a PCTN-MST é formada através da geração da PCTN a partir da base de dados e pela aplicação do limiar de influência $k = 17,5$. Em seguida, o filtro baseado em MST é aplicado sobre a rede formada pela PCTN. A rede resultante apresentou **174 correlações** e **177 métricas**. Logo, o número de métricas e também de correlações da PCTN-MST é menor que os da MST. Além disso, a PCTN-MST apresenta um compromisso melhor entre a quantidade de falhas detectadas e o número de falso-positivos, conforme apresentaremos na última parte da avaliação.

4.2.2. Estabilidade das Correlações

Apesar dos modelos matemáticos gerados a partir das redes de monitoramento terem sido criados de correlações identificadas como estáveis, é importante avaliar a estabilidade desses modelos. Um modelo é considerado estável se as relações invariantes persistem em diferentes execuções do sistema livre de falhas. Para avaliar a estabilidade das correlações geradas, submetemos os modelos a testes livres de falhas e observamos a quantidade que passa nos testes. Nessa avaliação, realizamos 20 testes para cada valor de PRDP, o qual varia de 0% a 90%. Todas as redes obtiveram seus melhores resultados dentro do intervalo de PRDP entre 1% e 3%, portanto, nos restringiremos a essa faixa de valores.

A Tabela 3 apresenta o número de modelos aceitos e a quantidade de métricas referenciadas pelos mesmos. É possível observar que o número de modelos, de todas as

redes, tende a aumentar de maneira significativa à medida que o PRDP aumenta. Por outro lado, a variação do PRDP tem pouco impacto na quantidade de métricas monitoradas. Além disso, com exceção da MST-PCTN, todas as redes possuem uma quantidade de métricas na mesma ordem de grandeza, ilustrando a dificuldade em encontrar soluções para reduzir o custo de monitoramento.

Rede de monitoramento	Correlações			Métricas		
	PRDP			PRDP		
	1%	2%	3%	1%	2%	3%
Pearson	340	735	900	182	184	184
PCTN	123	256	301	159	161	162
MST	55	117	135	170	175	175
MST-PCTN	16	33	44	40	41	41
PCTN-MST	48	112	129	158	160	161

Tabela 3. Quantidade de correlações e métricas após os testes de estabilidade.

4.2.3. Detecção de Falhas e Falso-Positivos

Para avaliar a capacidade de detecção de falhas dos modelos selecionados pelas redes de monitoramento, escolhemos algumas falhas típicas de sistemas computacionais [Wang et al. 2014] e as injetamos em nosso ambiente de testes. As falhas injetadas podem ser classificadas em: falha de software ou de programação, falha de desempenho, falha de configuração e falha de comunicação. Realizamos 31 experimentos com duração de 30 minutos cada. Em cada experimento, injetamos uma única falha, aproximadamente na metade do tempo do teste. A única exceção é a falha de configuração, a qual foi inserida logo no início do experimento. Para avaliar os falso-positivos, realizamos 31 experimentos livres de falhas. Conforme comentado previamente, os resultados se concentram na faixa de PRDP com os melhores desempenhos, isto é, entre 1% e 3%.

As Figuras 3 e 4 apresentam, respectivamente, o percentual de falhas detectadas e a taxa de falso-positivos das redes avaliadas. A rede de Pearson é capaz de detectar 100% das falhas, porém possui uma taxa de falso-positivos muito alta, próxima a 84% em sua melhor parametrização de PRDP. Além disso, é importante lembrar que a rede de Pearson possui o maior número de métricas monitoradas (184). Em outro extremo, está a MST-PCTN que, apesar de ser muito eficiente em termos de métricas monitoradas (40) e apresentar a menor taxa de falso-positivos, também exibe um percentual de detecção de falhas muito baixo, em torno de 20%. As outras redes operam fora desses extremos e há compromissos a serem analisados, conforme discutimos a seguir.

O melhor resultado da MST é uma detecção de falhas em torno de 70% e uma taxa de falso-positivos ligeiramente inferior a 50%. A PCTN detectou aproximadamente 84% de falhas e um pouco menos de 50% de taxa de falso-positivos. Ou seja, um desempenho superior à MST, o qual se torna ainda mais relevante por vir a um menor custo de monitoramento, uma vez que são usadas 162 métricas pela PCTN contra 184 da MST. Por fim, temos a PCTN-MST que tem praticamente a mesma quantidade de métricas que a PCTN e apresenta uma elevação no percentual de falhas detectadas (80%). No entanto, a PCTN-MST exibe uma taxa menor de falso-positivos (40%). Assim, a PCTN-MST alcança o

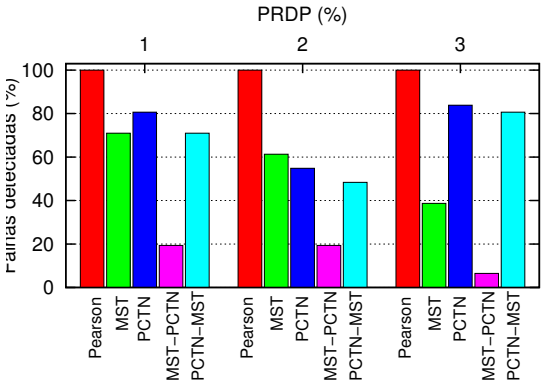


Figura 3. Porcentagem de falhas detectadas em função da PRDP.

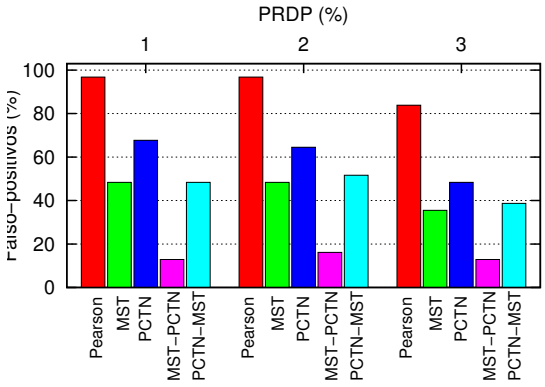


Figura 4. Taxa de falso-positivos em função da PRDP.

melhor compromisso e supera a MST: 1) reduzindo em 8% o número de métricas monitoradas, 2) aumentando em 10% o percentual de falhas detectadas e 3) reduzindo também em 10% a taxa de falso-positivos. Naturalmente, esses valores podem ser diferentes em outros cenários, mas os benefícios da abordagem PCTN-MST tendem a persistir.

5. Conclusão e Trabalhos Futuros

Neste trabalho, introduzimos a correlação parcial para selecionar correlações e métricas monitoradas que são empregadas na detecção de falhas em aplicações *Web* multi-camadas. A redução da quantidade de correlações e métricas, desde que oferecendo acurácia adequada, traz benefícios através da redução do consumo de recursos de centros de dados. Através de experimentos em um ambiente de testes real, mostramos que uma de nossas abordagens consegue superar uma estratégia estado-da-arte, gerando uma rede de correlações monitoradas menor e com acurácia superior. Em nossa abordagem, as correlações foram extraídas a partir de procedimentos *offline*. No entanto, em ambientes que apresentem carga de trabalho com alta dinamicidade, estratégias para detectar mudanças relevantes na carga de trabalho e a atualização dos modelos de forma *online* podem trazer benefícios e serão estudadas em trabalhos futuros.

Agradecimentos

Este trabalho foi parcialmente financiado por FAPEG, RNP e CNPq.

Referências

Aceto, G., Botta, A., De Donato, W., and Pescapè, A. (2013). Cloud monitoring: A survey. *Computer Networks*, 57(9):2093–2115.

Baba, K., Shibata, R., and Sibuya, M. (2004). Partial correlation and conditional correlation as measures of conditional independence. *Australian & New Zealand Journal of Statistics*, 46(4):657–664.

Bezenek, T., Cain, T., Dickson, R., Heil, T., Martin, M., McCurdy, C., Rajwar, R., Weglarz, E., Zilles, C., and Lipasti, M. (2014). Java TPC-W Implementation Distribution. `http://pharm.ece.wisc.edu/tpcw.shtml`. [Último acesso: 21-Set-2014].

Chen, H., Jiang, G., Yoshihira, K., and Saxena, A. (2010). Invariants based failure diagnosis in distributed computing systems. In *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, pages 160–166.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 2nd edition edition.

De La Fuente, A., Bing, N., Hoeschele, I., and Mendes, P. (2004). Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20(18):3565–3574.

Forster, F. (2014). collectd. `http://collectd.org`. [Último acesso: 05-Dez-2014].

IBM (2014). IBM SmartCloud Analytics. `http://www-03.ibm.com/software/products/en/ibm-smartcloud-analytics---predictive-insights`. [Último acesso: 06-Dez-2014].

Jiang, G., Chen, H., and Yoshihira, K. (2006). Modeling and tracking of transaction flow dynamics for fault detection in complex systems. *IEEE Trans. Dependable Secur. Comput.*, 3(4):312–326.

Jiang, M., Munawar, M. A., Reidemeister, T., and Ward, P. A. (2009). System monitoring with metric-correlation models: problems and solutions. In *Proceedings of the 6th international conference on Autonomic computing*, pages 13–22.

Kenett, D. Y., Tumminello, M., Madi, A., Gur-Gershgoren, G., Mantegna, R. N., and Ben-Jacob, E. (2010). Dominating clasp of the financial sector revealed by partial correlation analysis of the stock market. *PLoS ONE*, 5(12).

Magalhaes, J. and Silva, L. (2010). Detection of performance anomalies in web-based applications. In *Network Computing and Applications (NCA), 2010 9th IEEE International Symposium on*, pages 60–67.

Menasce, D. (2002). Tpc-w: a benchmark for e-commerce. *Internet Computing, IEEE*, 6(3):83–87.

Mi, N., Casale, G., Cherkasova, L., and Smirni, E. (2010). Sizing multi-tier systems with temporal dependence: benchmarks and analytic models. *Journal of Internet Services and Applications*, 1(2):117–134.

Microsoft (2013). System Center Operations Manager. `http://technet.microsoft.com/en-us/systemcenter/bb497976`. [Último acesso: 06-Dez-2014].

Munawar, M. A., Jiang, M., Reidemeister, T., and Ward, P. A. (2009). Filtering System Metrics for Minimal Correlation-Based Self-Monitoring. In *Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 233–242.

Peiris, M., Hill, J. H., Thelin, J., Bykov, S., Kliot, G., and Konig, C. (2014). Pad: Performance anomaly detection in multi-server distributed systems. In *7th IEEE International Conference on Cloud Computing (IEEE Cloud 2014)*.

Simic, B. (2014). Trac research. `http://www.new.trac-research.com`. [Último acesso: 10-Ago-2014].

Wang, C., Kavulya, S. P., Tan, J., Hu, L., Kutare, M., Kasick, M., Schwan, K., Narasimhan, P., and Gandhi, R. (2013). Performance troubleshooting in data centers: An annotated bibliography? *SIGOPS Oper. Syst. Rev.*, 47(3):50–62.

Wang, T., Wei, J., Zhang, W., Zhong, H., and Huang, T. (2014). Workload-aware anomaly detection for web applications. *The Journal of Systems and Software*, 89:19–32.

Zhang, Q., Cherkasova, L., and Smirni, E. (2007). A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications. In *Proceedings of the Fourth International Conference on Autonomic Computing*, pages 27–36.

Um Protocolo MAC de Cooperação para Redes PLC

Roberto M. Oliveira[1,2], **Michelle S. P. Facina**[2], **Moises V. Ribeiro**[2,3], **Alex B. Vieira**[1]

[1]Departmento de Ciência da Computação
Universidade Federal de Juiz de Fora – Juiz de Fora, MG – Brasil.

[2]Departmento de Engenharia Elétrica
Universidade Federal de Juiz de Fora – Juiz de Fora, MG – Brasil.

[3]Smarti9 Ltda. - Juiz de Fora, Brazil

{rmassi.oliveira,michelle.soares,mribeiro,}@engenharia.ufjf.br

alex.borges@ufjf.edu.br

Abstract. *This paper introduces medium access control (MAC) protocol improving the performance of in-home power line communication (PLC) systems when a cooperative single relay model is considered. The proposed protocol is able to work regardless of the idleness or busyness of the relay node to cooperate with the source node at the link layer level. To support this proposal, we briefly present a statistical analysis of packet loss and goodput performances that are obtained with an uncoded orthogonally frequency division multiplexing scheme when estimates of measured in-home cooperative PLC channels are considered. Numerical results, when an orthogonal frequency division multiple access-time division multiple access scheme, reveal in which conditions the proposed protocol can improve PLC system performance at the link layer. Overall, the proposed protocol is capable of reducing the packet loss in half and increasing up the goodput in* 46%.

Resumo. *Este artigo apresenta um protocolo de cooperação de controle de acesso ao meio (MAC) que melhora o desempenho de redes domésticas de sistemas de comunicação via rede elétrica (PLC) considerando um modelo com um único relay. O protocolo proposto é capaz de operar, independentemente, da ociosidade ou da ocupação do relay, promovendo cooperação com o nó fonte no nível da camada de enlace. Para apoiar essa proposta, nós apresentamos uma análise estatística da perda de pacotes e do goodput obtidos com um esquema de acesso múltiplo por divisão na frequêcia não codificado (uncoded OFDM), considerando medidas realizadas em canais PLC cooperativos de ambientes domésticos. Os resultados numéricos, quando realizado o acesso múltiplo por divisão na frequência e o acesso múltiplo por divisão no tempo (OFDMA-TDMA), revelam em quais condições o protocolo proposto melhora o desempenho de um sistema PLC na camada de enlace. Na média, o novo mecanismo proposto reduz as perdas pela metade e melhora o goodput em* 46%.

1. Introdução

Atualmente, há uma grande discussão em torno da utilização de um protocolo de cooperação adequado para melhorar desempenho da comunicação via rede elétrica

(PLC). A maior parte das contribuições nessa área discutem os tipos de melhorias que podem ser obtidos 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*, em camadas superiores [Noori and Lampe 2013, Bilbao et al. 2014]. Seja na camada física, ou em camadas superiores, técnicas de cooperação têm o potencial de melhorar a confiabilidade da rede (e.g. redução da perda de pacotes) e de aumentar seu desempenho (e.g. ganhos em *goodput*). Mais ainda, as técnicas existentes em várias camadas podem ser combinadas com o objetivo de mitigar as barreiras na comunicação de dados oferecidas pela rede elétrica.

A literatura revela uma lacuna com relação a trabalhos de cooperação em sistemas PLC. Há também uma falta de conhecimento sobre o comportamento de tais sistemas na prática. Por exemplo, não conhecemos como é a taxa de erro de pacotes na camada de enlace de sistemas PLC. Consequentemente, são desconhecidas as circunstâncias nas quais a cooperação nessa camada poderia melhorar o desempenho de redes PLC.

Nesse sentido, neste trabalho propomos um mecanismo adaptativo de cooperação na camada MAC. Tradicionalmente, a cooperação explora a ociosidade de um nó cooperador (*relay*) [Lee et al. 2013] ou, simplesmente, a diversidade de canais (ou enlaces) existentes [Yang et al. 2010]. A nova abordagem que propomos explora ambos os fatores, associa a diversidade de caminhos a um esquema de acesso múltiplo de divisão de frequência ortogonal e acesso múltiplo por divisão de tempo (OFDMA-TDMA). Assim, nosso mecanismo de cooperação pode tanto melhorar o *goodput* como reduzir a perda de pacotes na rede.

Para demonstrar tais melhorias, fizemos simulações utilizando estimativas de taxa de erro de pacote em um modelo com um único *relay*. 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. Como resultados, mostramos em quais circunstâncias a cooperação na camada de enlace pode ser útil ao sistema PLC. Os resultados obtidos mostram que, quando comparado a um sistema sem cooperação, o novo protocolo é capaz de aumentar o *goodput* em 46%. Mais ainda, sistemas sem cooperação apresentam, em média, 115%" a mais de perdas quando comparados com o novo protocolo.

Em suma, nossas contribuições se dividem em duas etapas: primeiro, desenvolvemos um protocolo adaptativo de cooperação na camada de enlace redes PLC; finalmente, utilizamos o protocolo desenvolvido para delimitar os cenários nos quais a cooperação na camada de enlace é viável e oferece melhoras no desempenho de redes PLC.

2. Formulação do Problema

A Figura 1 mostra uma rede PLC doméstica em um modelo com um único *relay*. Nessa rede, S, R e D equivalem aos nós fonte, *relay* e destino, respectivamente. Tal cenário é composto por três enlaces: fonte-destino (SD), fonte-*relay* (SR) e *relay*-destino. Além disso, cada canal possui taxas de erro de pacote independentes denominadas por PER_i, $i \in \{SD, SR, RD\}$. Apesar de haver apenas um *relay* nesse cenário, a proposta atual pode ser expandida ao uso de múltiplos *relays*. Além disso, espera-se que o uso de um

único *relay* seja um limite inferior de possíveis ganhos pelo uso de cooperação na camada de enlace de redes PLC.

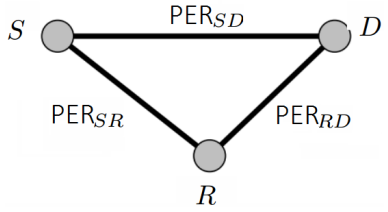


Figura 1. Modelo cooperativo com um único *relay*.

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.

Realizamos uma campanha de medição em sete residências de uma área típica urbana brasileira, a fim de estimar a PER dos enlaces SD, SR e RD. Assim, adquirimos estimativas de canais PLC cooperativos e ruídos aditivos em ambientes domésticos. Para realizar as medições, utilizamos um equipamento composto por dois computadores, placas de geração e aquisição de dados e acopladores capacitivos, cujos detalhes podem ser vistos em [Colen et al. 2013]. Além disso, foi usado um sistema baseado em escuta para estimar a resposta em frequência dos canais, como visto em [Oliveira et al. 2014]. Nessa campanha, coletamos 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.

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).

A Figura 2 apresenta as posições comumente encontradas durante nossas medições do nó R, em relação aos nós S e D. Intuitivamente, nos casos #1 e #3, os enlaces SR e RD são menores que o enlace direto. Assim, esperamos que a transmissão direta pelo enlace SD apresente uma taxa de erros maior que os enlaces indiretos. Nesses dois casos, a cooperação pode melhorar a confiabilidade do sistema e melhorar seu desempenho. Por outro lado, os casos #2 e #4 apresentam enlaces indiretos com distancias tão grandes quanto o enlace direto (ou maiores como no caso #4). Em ambos os casos, a utilização do *relay* como cooperador pode trazer benefícios limitados ao sistema.

A Figura 3 mostra valores típicos de PERs conforme variamos a potência dos canais. Todas as PER tendem a 1 quando a potência total é reduzida; e a 0 quando há um aumento da mesma. Além disso, para todo o conjunto de potências, observamos que as PER de SD são maiores que as obtidas para os enlaces SR e RD. Para 10 dBm, por exemplo, notamos que os valores de PER para o enlace SD são de até 60%, enquanto que

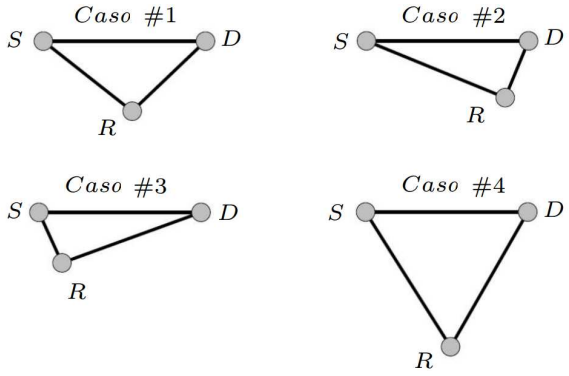


Figura 2. Posições relativas de R obtidas na campanha de medição.

para os enlaces SR e RD, esses valores são de 40% e 20%, respectivamente. Portanto, o aproveitamento de um *relay* para auxiliar o nó fonte na camada MAC, com o objetivo de melhorar a performance da rede, é um fator interessante investigado neste trabalho.

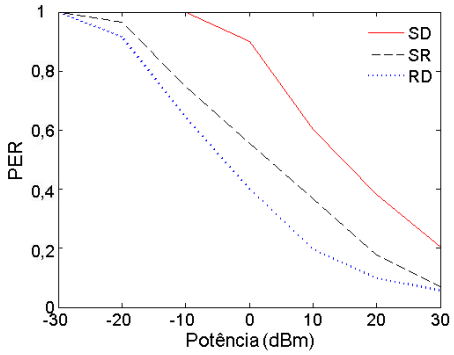


Figura 3. PER média para cada enlace com variação da potência.

Adicionalmente, a Figura 4 mostra a função de distribuição cumulativa (CDF) dos valores relativos de perda de pacotes (PER_{SR}/PER_{SD} e PER_{RD}/PER_{SD}), denotados por ρ. Nesse caso, consideramos todos os canais medidos com a potência total de 10 dBm. Como podemos observar, as probabilidades para que PER_{SR} e PER_{RD} sejam menores que PER_{SD} ($\rho < 1$), são de 0.83 e 0.71, respectivamente. Isso significa que os enlaces SR e RD, que compõem um caminho alternativo entre S e D, possuem erros proporcionais menores que o caminho SD. Então, a comunicação através do *relay* apresentará menos problemas que o envio direto ao destino.

Por fim, é importante ressaltarmos que, nas próximas seções, nossas análises se limitam a um sistema com um único *relay*, pois as medições realizadas se limitam a esse cenário. No entanto, os protocolos analisados podem ser utilizados em sistemas com n-*relays* de forma transparente.

3. Protocolo de cooperação na camada MAC para redes PLC

Como mostramos na Figura 1, redes PLC possuem características de um barramento. Mais precisamente, um pacote enviado pela origem S é recebido tanto pelo *relay* R, quanto pelo destino D. Nessa situação, pode-se assumir que os nós envolvidos possuem *buffer*

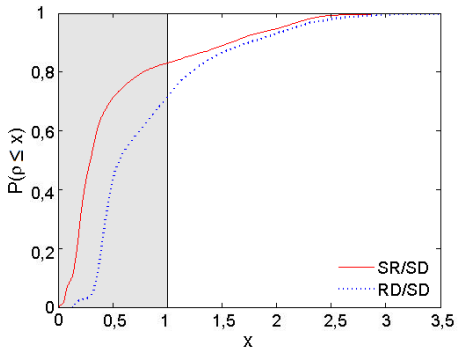


Figura 4. Valor relativo de perda de pacotes quando a potência é de 10 dbm.

suficientemente adequado para o armazenamento temporário de pacotes. Assim, caso D envie uma confirmação negativa (NACK)[1], tanto S quanto R podem tentar retransmitir os pacotes correspondentes utilizando seus próprios recursos. Nesse esquema, o nó D é responsável pela detecção de pacotes errados ou perdidos. Caso algo do tipo ocorra, D envia uma mensagem de NACK em *broadcast*, requisitando a retransmissão de um pacote específico.

Neste trabalho, consideramos um esquema TDMA para acesso à rede elétrica por parte dos usuários. Além disso, assumimos que um *frame* é dividido entre períodos de sinalização (SP) e de dados (DP). O período de sinalização SP é usado para a troca de mensagens de controle, as quais ajustam detalhes da transmissão e da cooperação. Já o período de dados DP é alocado para a transmissão de pacotes de dados. Adicionalmente, os nós S, R e D, nessa ordem, possuem *time-slots* alocados sucessivamente.

O protocolo proposto para cooperação na camada MAC (PLC-CMAC) explora a disponibilidade do *relay* para a retransmissão de dados. Apesar de analisarmos o protocolo PLC-CMAC utilizando um único *relay*, é possível aplicá-lo a um sistema com múltiplos *relays* de forma transparente. A Figura 5 ilustra o funcionamento geral do protocolo de cooperação proposto. O processo é iniciado em DP (períodos de dados), quando o nó S envia um dado para o nó D, que também é recebido por R. Tanto o *relay* R quanto a origem armazenam temporariamente o pacote (PKT) enviado. Caso haja uma confirmação positiva (ACK), ambos (S e R) descartam esse pacote e o nó S poderá enviar um novo dado em seu próximo *time-slot*. Caso haja uma confirmação negativa (NACK) e o *relay* esteja ocioso (não possua quaisquer pacotes para transmitir em seu *time-slot*), o mesmo envia uma mensagem *want to cooperate* (WTC) para o nó S oferecendo seu próprio *time-slot* para cooperação. Portanto, caso S receba uma mensagem WTC, a técnica dependente da ociosidade do *relay*, proposta em [Lee et al. 2013], é aplicada para realizar a retransmissão do pacote. Essa técnica foi desenvolvida para redes sem fio e é nomeada como *Dynamic Slot Assignment Cooperation* (DSAC).

A Figura 6 ilustra o funcionamento da técnica DSAC. Essa imagem mostra dois *frames* consecutivos. Durante o primeiro, em DP, o nó S envia um dado (PKT1) para o D, que também é recebida por R, conforme mencionado anteriormente. Então, o nó D detecta um erro de pacote e responde com uma mensagem NACK em *broadcast*. No

[1]Por exemplo, em canais PLC, quadros podem ser danificados por interferências e atenuações do sinal.

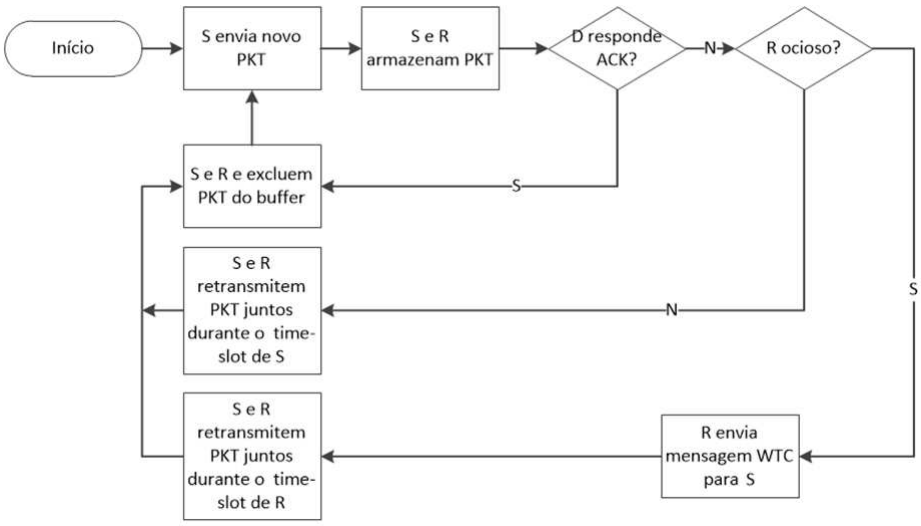


Figura 5. Protocolo MAC de cooperação para redes PLC domésticas.

próximo frame SP, o nó R envia ao S a mensagem WTC, oferecendo seu próprio *time-slot* para cooperação. O nó S poderá enviar um novo pacote (PKT2) em seu *time-slot* e retransmitir o PKT1 juntamente com o *relay* durante o *time-slot* de R. Essa retransmissão, realizada a partir de dois enlaces distintos (SD e RD), explora a vantagem da diversidade de caminhos para melhorar o desempenho do sistema em termos de redução na perda de pacotes. Além disso, como não há a necessidade de espera até que a fonte S possa usar seu próprio *time-slot* para retransmissões, pode haver um ganho em *goodput*.

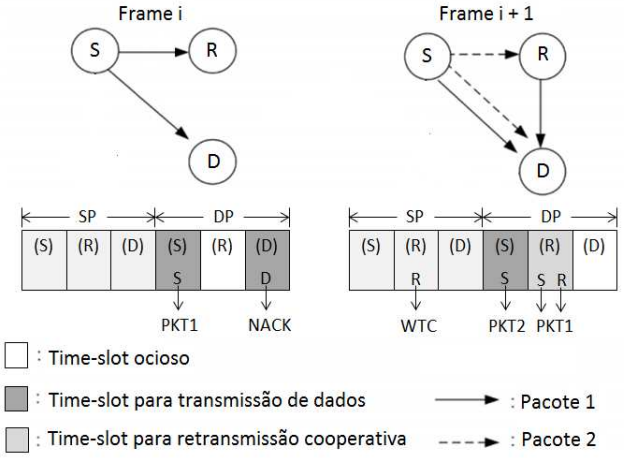


Figura 6. Modo de operação do protocolo de cooperação quando R está ocioso [Lee et al. 2013].

Caso o nó R não esteja ocioso para oferecer seu próprio *time-slot* para retransmitir, o DSAC funciona sem cooperação, como ocorreria de qualquer forma. Porém, para evitar esse problema, o PLC-CMAC utiliza o protocolo independente da ociosidade do *relay*, no qual a cooperação é realizada no *time-slot* do nó S. Essa técnica, apresentada na Figura 7, foi discutida em [Yang et al. 2010] para redes sem fio e é nomeada como *Cooperative Diversity Protocol* (CD).

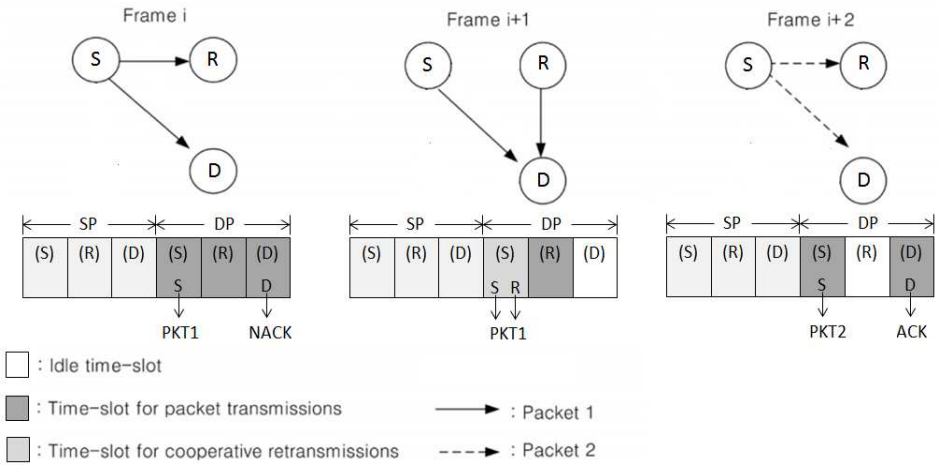


Figura 7. Modo de operação do protocolo de cooperação quando R está ocupado [Yang et al. 2010].

Na Figura 7, observamos que o nó R não envia a mensagem WTC para o nó S durante o segundo *frame*. Isso significa que não poderá haver cooperação no *time-slot* do *relay*. Sendo assim, o nó S não será capaz de transmitir um novo pacote (PKT2) em seu *time-slot*, pois PKT1 precisa ser retransmitido junto com o *relay* no *time-slot* de S. Portanto, o CD pode não resultar em um ganho de *goodput* tão bom quanto o DSAC quando o *relay* está ocioso. Para o exemplo dado, enquanto DSAC gasta dois *frames* para a transmissão de dois pacotes distintos, CD precisará de três. No entanto, o uso de CD pode manter o ganho de diversidade, melhorando a redução na perda (erros) de pacotes ao manter a retransmissão por dois caminhos distintos (SD e RD).

A partir das descrições feitas ao funcionamento do PLC-CMAC, podemos notar que o mesmo é a fusão de dois protocolos distintos de cooperação na camada MAC, que foram desenvolvidos para redes sem fio. Essa junção resulta na utilização oportunista do nó R na cooperação com o nó S. Visto isso, o protocolo proposto pode oferecer as seguintes vantagens:

- Caso o nó R esteja ocioso e uma retransmissão seja necessária, essa será executada durante o *time-slot* do *relay*. Assim, o *goodput* da rede será melhorado uma vez que a fonte poderá transmitir um novo pacote no próximo momento. Além disso, a perda de pacotes será reduzida devido ao uso de dois caminhos distintos com diferentes PER (PER_{SD} e PER_{RD}) para retransmissão.
- Caso o nó R esteja ocupado e uma retransmissão seja necessária, a mesma será realizada durante o *time-slot* do nó S. Portanto, o *goodput* da rede não será melhorado como na caso anterior, visto que a fonte terá que gastar seu próprio *time-slot* para retransmitir pacotes. Apesar disso, essa técnica é capaz de manter uma boa redução na perda de pacotes em retransmissões, devido à diversidade de caminhos utilizado no momento de cooperação.

4. Avaliação

Nesta seção, apresentamos simulações que mostram a eficiência do protocolo PLC-CMAC proposto. Nesse sentido, foram considerados os 4 cenários mais comuns em redes

PLC, mostrados na Figura 2. De forma sucinta, em uma rede PLC doméstica, o *relay* pode estar situado aproximadamente equidistante da fonte e do destino, mais próximo da fonte que do destino, mais próximo do destino do que da fonte e, em um caso extremo, longe tanto da fonte quanto do destino. Intuitivamente, a distância é relacionada a taxa de erros do canal, o que confirmamos com as caracterizações apresentadas na Seção 2.

Consideramos também um sistema *uncoded* OFDMA-TDMA, no qual metade das subportadoras (as pares) são alocadas para o nó S e a outra metade (as ímpares), para o nó R, quando ocorre cooperação. Caso não haja cooperação, o OFDMA aloca todas as subportadoras para um nó quando o mesmo está usando seu próprio *time-slot* para a transmissão de pacotes. Adotamos a modulação digital BPSK e as potências $P \in \{-20, -10, 0, 10, 20, 30\}$ dBm, sendo que $P_0 = P_1 = P/2$ são igualmente alocadas entre todas as subportadoras. A frequência de banda utilizada é de 1.7 até 100 MHz.

Além disso, a análise de desempenho é quantificada em termos de perda de pacotes e *goodput*. A primeira é definida como o número de perdas dividido pelo número de pacotes enviados. O segundo, é definido pelo número de recepções corretas pelo nó D em um dado período de tempo. Nós variamos a taxa de ociosidade do *relay* em saltos de 25%. Quando não mencionado, assumimos que a ociosidade do *relay* ocorre em 50% de seus *time-slots*. Esse número reflete um bom compromisso para ilustrar o funcionamento do nosso protocolo.

Para a obtenção dos valores de taxa de perda de pacotes e *goodput*, reproduzimos o funcionamento dos protocolos, descritos na Seção 3, no software MATLAB[2]. Para simular a transmissão de dados, consideramos apenas a PER[3] calculada para cada enlace do modelo adotado. Sendo assim, ignoramos parâmetros como taxa de transferência e atrasos da rede. Além disso, consideramos as mensagens de controle (ACK, NACK e WTC) imunes a erros nos testes executados.

Realizamos 11618 iterações no simulador, as quais correspondem a um conjunto de erros de pacote medidos para cada um dos três enlaces observados na Figura 1. Essas simulações cobrem todas as amostras de todos os casos mostrados na Figura 2. Para cada interação simulada, executamos o envio de 100 pacotes de dados de S para D, onde cada falha no envio, quando ocorre, gera uma única tentativa de retransmissão. Nesse caso, a falha na retransmissão significa a perda do dado enviado. Finalmente, para garantir que todos os 100 pacotes da lista de S fossem transmitidos, utilizamos um período de 200 *frames*.

A Figura 8 mostra a função de distribuição acumulada (CDF) da relação de perdas existente em um sistema PLC tradicional, comparado ao um sistema com o uso da proposta de cooperação. Avaliamos sistemas com diferentes potências de funcionamento (de -20 dBm a 30 dBm). A partir do gráfico, notamos que o ganho relativo em potências muito baixas é pequeno. Por exemplo, para -20 dBm, os ganhos relativos pertencem à curva mais deslocada à esquerda (menores valores). Isso ocorre, pois, em baixas potências, a taxa de erros costuma ser muito grande em todos os enlaces. Dessa forma, o *relay* não tem a oportunidade de cooperar, uma vez que seu enlace com o destino também apresenta altos valores de erros.

[2]Os algoritmos para simulação estão disponíveis no site http://netlab.ice.ufjf.br/plc/.

[3]As amostras utilizadas nas simulações estão disponíveis no site http://netlab.ice.ufjf.br/plc/.

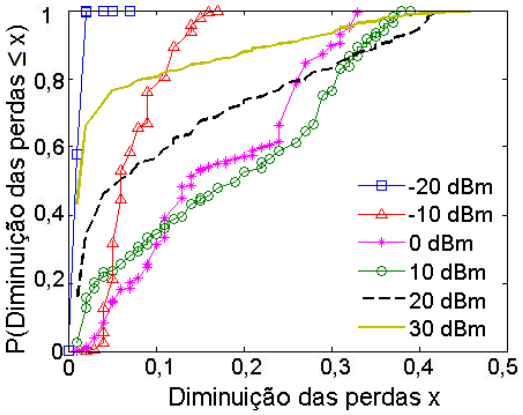


Figura 8. CDFs da redução da perda de pacotes com a variação da potência.

Para potências altas, e.g. 30 dBm, temos uma situação oposta. Os canais praticamente não apresentam erros (ou apresentam taxas muito baixas). Dessa forma, a transmissão direta entre origem e destino ocorre praticamente sem erros, o que não gera oportunidades de cooperação para o *relay*.

As potencias intermediárias são mais adequadas ao nosso cenário de cooperação. De fato, de acordo com o que apresentamos na Figura 8, observamos que essas curvas tendem a estar deslocadas para a direita, indicando uma melhor taxa de melhora. Nesse sentido, adotamos valores de potência de 10 dBm avaliar o protocolo de cooperação proposto com as medidas existentes.

A Figura 9 mostra a melhora do *goodput* na aplicação do PLC-CMAC. Novamente, os melhores ganhos são verificados para níveis intermediários de potência, o que revela uma correspondência de comportamento entre o aumento do *goodput* e a redução das perdas. Para uma potência de 10 dBm, mais da metade dos casos apresentou ganhos de pelo menos 20% no *goodput*. Nesse mesmo cenário, as perdas diminuem significativamente, apresentando pelo menos 35% de redução.

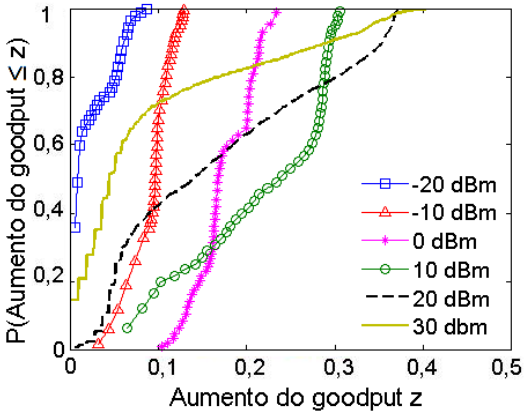


Figura 9. CDFs do aumento no *goodput* com a variação da potência.

As Figuras 10 e 11 comparam o novo protocolo PLC-CMAC com as abordagens tradicionais DSAC e CD (utilizadas em meio não cabeado). Nessas comparações, utilizamos a potência de 10 dbm para apresentar o cenário com melhores possibilidades de

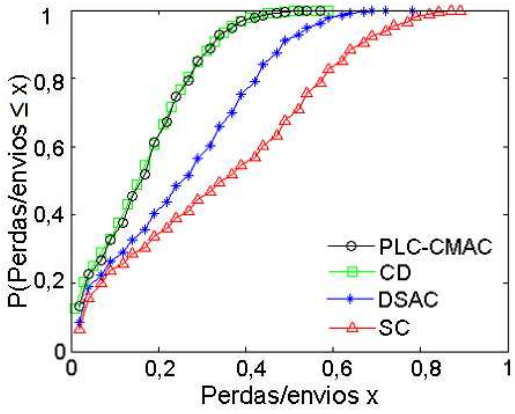


Figura 10. CDFs da perda de pacotes para a potência é de 10 dBm.

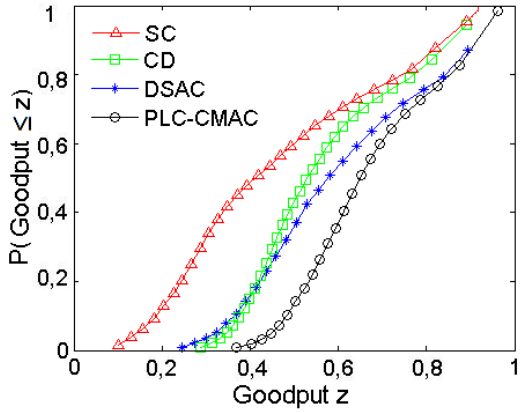


Figura 11. CDFs do *goodput* para a potência é de 10 dBm.

cooperação, conforme discutido anteriormente.

De acordo com a Figura 10, o novo protocolo atinge os mesmos resultados de perda de pacote que o esquema CD. Nesse caso, tanto PLC-CMAC quanto no protocolo CD, as cooperação é sempre realizada, independente da ociosidade do *relay*. Mais ainda, o reenvio é sempre feito pelos enlaces RD e SD, o que gera uma diversidade de caminhos equivalente para ambos os protocolos. O método sem cooperação não faz uso de nenhum caminho alternativo para retransmissões e, portanto, representa o pior caso. Enquanto o PLC-CMAC e o CD oferecem taxas de perda de pacotes inferiores a 0,35 em 90% dos resultados obtidos, o SC e o DSAC resultam nas taxas de até a 0,6 e 0,5, respectivamente. Nesse contexto, um sistema sem cooperação tem taxas de erro 115% maiores quando comparados a um sistema que utiliza o PLC-CMAC.

A Figura 11 também mostra altos ganhos no uso do novo protocolo. Enquanto o CD apresenta taxas de erros tão baixas quando o PLC-CMAC, seu *goodput* não é tão bom. Isso ocorre pois o novo protocolo explora a ociosidade do *relay* para fazer retransmissões. Assim, não há a necessidade de espera pelo *slot* de tempo da origem. Tentativas de retransmissão são realizadas mais rapidamente, melhorando, por consequência, o *goodput*.

Nesse mesmo cenário, o protocolo DSAC apresenta taxas de *goodput* superiores ao CD. Mesmo assim, o novo protocolo supera o DSAC. Isso acontece, pois, enquanto o

DSAC não coopera quando R está ocupado, o PLC-CMAC funciona tal qual o protocolo CD. Consequentemente, o novo protocolo apresenta valores de *goodput* superiores aos obtidos pelas técnicas tradicionais utilizadas em meio sem fio. Em detalhes, enquanto 50% dos casos tem taxas de *goodput* superiores a 0,65; em um sistema sem cooperação, 50% dos casos não atingem taxas de 0,3. Na média, o *goodput* oferecido pelo novo protocolo é 46% melhor que o obtido no caso sem cooperação.

Finalmente, nós avaliamos o impacto do nível de ociosidade do *relay* em cada protocolo estudado. As tabelas 1 e 2 mostram valores médios de perda de pacote e *goodput* com 99% de intervalo de confiança.

Tabela 1. Média da perda de pacotes em função da variação da ociosidade de R.

Protocolo/ Ociosidade	0%	25%	50%	75%	100%
SC	0,3448± 0,0071	0,3448± 0,0071	0,3442± 0,0071	0,3448± 0,0071	0,3448± 0,0071
DSAC	0,3447± 0,0071	0,2988± 0,0061	0,2533± 0,0052	0,2090± 0,0043	0,1640± 0,0035
CD	0,1634± 0,0035	0,1634± 0,0035	0,1641± 0,0035	0,1634± 0,0035	0,1634± 0,0035
PLC-CMAC	0,1638± 0,0035	0,1636± 0,0035	0,1633± 0,0035	0,1632± 0,0035	0,1634± 0,0035

Tabela 2. Média do *goodput* em função da variação da ociosidade de R.

Protocolo/ Ociosidade	0%	25%	50%	75%	100%
SC	0,4663± 0,0071	0,4663± 0,0071	0,4670± 0,0071	0,4663± 0,0071	0,4663± 0,0071
DSAC	0,4666± 0,0071	0,5309± 0,0066	0,6100± 0,0060	0,7078± 0,0050	0,8360± 0,0035
CD	0,5741± 0,0052	0,5741± 0,0052	0,5737± 0,0052	0,5741± 0,0052	0,5741± 0,0052
PLC-CMAC	0,5735± 0,0052	0,6211± 0,0050	0,6767± 0,0047	0,7469± 0,0042	0,8366± 0,0035

Notamos que, para 0% de ociosidade do *relay*, o DSAC funciona como o SC e o PLC-CMAC, como o CD (vide Figura 5 para o caso de *relay* ocupado). Quando a ociosidade aumenta (25%, 50% e 75%), percebemos uma melhora nos resultados de DSAC, visto que o mesmo é um protocolo dependente da ociosidade do *relay*. No entanto, o CD não apresenta variações, pois se trata de um protocolo independente da ociosidade do nó R. Independente da disponibilidade de R, o PLC-CMAC resulta nas menores probabilidades de perda de pacote e nos mais altos *goodputs* entre todos as técnicas avaliadas. Por último, vale destacar que, quando R apresenta 100% de ociosidade, o PLC-CMAC opera como o DSAC (vide Figura 5 para o caso de *relay* ocioso), e, portanto, ambos apresentam os mesmos resultados.

Observamos então que, enquanto as abordagens tradicionais apresentam ganhos em uma das métricas de interesse isoladamente, o novo protocolo apresenta ganho em

ambos. Nesse caso, ele é equivalente ao melhor caso de funcionamento de qualquer uma das abordagens anteriores.

5. Conclusões

Nesse artigo, nós propomos um protocolo de cooperação adaptativo na camada MAC para redes PLC domésticas que faz uso de um modelo com um único *relay* e de um esquema OFDMA-TDMA. O novo protocolo é capaz de operar independentemente da ociosidade ou da ocupação do *relay*, promovendo cooperação com o nó fonte no nível da camada de enlace. Nós apresentamos também dados provenientes de uma campanha de medição, a qual provê estimativas de taxas de erro de pacote em um esquema HS-OFDM. Esses dados são relevantes para a avaliação de mecanismos de cooperação em redes PLC.

Nossas simulações mostram que o uso de potências intermediárias, normalmente encontradas na prática, propiciam um meio favorável a cooperação em redes PLC. Por exemplo, para potências de 10 dbm, tanto perdas quanto o *goodput* são favorecidos. Mais ainda, as 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*. Por exemplo, enquanto o PLC-CMAC oferece taxas de perda de pacotes inferiores a 0,35 em 90% dos resultados obtidos, um sistema sem cooperação apresenta taxas de erro de até a 0,6. Na média, um sistema sem cooperação tem taxas de erro 115% superiores a um sistema que utiliza o PLC-CMAC. Além disso, o novo protocolo é capaz de aumentar o *goodput* em 46% na média.

Portanto, nesse trabalho, nós exploramos novas possibilidades de melhoria do desempenho com o uso de cooperação na camada de enlace de redes PLC. Além disso, mostramos que a cooperação oferece os melhores resultados de taxa de perda de pacote e *goodput* quando usada em um sistema com potências intermediárias.

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.

Colen, G. R., Marques, C. A. G., Oliveira, T. R., de Campos, F. P. V., and Ribeiro, M. V. (2013). Measurement setup for characterizing low-voltage and outdoor electric distribution grids for plc systems. In *Proc. PES Conference On Innovative Smart Grid Technologies Latin America*, pages 1–5.

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.

Lee, J., Noh, H., and Lim, J. (2013). A cooperative tdma mac protocol using dynamic slot assignment scheme. In *Proc. IEEE Conference on Information Networking*, pages 216–220.

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.

Oliveira, T. R., Marques, C. A. G., Finamore, W. A., Netto, S. L., and Ribeiro, M. V. (2014). A methodology for estimating frequency responses of electric power grids. *Journal of Control, Automation and Electrical Systems*, 25(6):720–731.

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.

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.

Yang, Z., Yao, Y., Li, X., and Zheng, D. (2010). A tdma-based mac protocol with cooperative diversity. *IEEE Communications Letters*, 14(6):542–544.

Uma Plataforma de Middleware para Integração de Dispositivos e Desenvolvimento de Aplicações em *e-health*

Pedro Maia[1], Augusto Baffa[2], Everton Cavalcante[1,3], Flavia C. Delicato[4], Thais Batista[1], Paulo F. Pires[4]

[1]DIMAp, Universidade Federal do Rio Grande do Norte – Natal, Brasil
[2]DI, Pontifícia Universidade Católica do Rio de Janeiro – Rio de Janeiro, Brasil
[3]IRISA-UMR CNRS/Université de Bretagne-Sud – Vannes, França
[4]DCC/PPGI, Universidade Federal do Rio de Janeiro – Rio de Janeiro, Brasil

{pedropetrovitch, augbaffa, evertonranielly, fdelicato, thaisbatista, paulo.f.pires}@gmail.com

Resumo. *Na visão da Internet das Coisas (IoT), qualquer objeto físico ou virtual pode ser identificado, endereçável, controlado e monitorado via Internet. No cenário de e-health, tais objetos inteligentes podem ser sensores corporais que permitem um monitoramento contínuo de sinais vitais de pacientes. Diante da inerente heterogeneidade desses dispositivos, é necessário garantir interoperabilidade entre eles e facilitar o desenvolvimento de aplicações que façam uso dos seus dados. Nesse contexto, apresentamos a EcoHealth (Ecossistema de Dispositivos de Health Care), uma plataforma de middleware que integra sensores corporais heterogêneos para permitir o monitoramento remoto de pacientes e melhoria de diagnósticos médicos. Neste artigo são apresentadas a arquitetura da EcoHealth e uma avaliação de seu desempenho considerando uma aplicação de e-health desenvolvida como prova de conceito.*

Abstract. *In the Internet of Things (IoT) vision, any physical or virtual object can be identified, addressable, controlled, and monitored via Internet. In the e-health scenario, these smart objects can be body sensors allowing the continuous monitoring of vital signs of patients. Due to the inherent heterogeneity of such devices, it is necessary to enable interoperability among them and to facilitate the development of applications that make use of the provided data. In this context, this paper introduces EcoHealth (Ecosystem of Health Care Devices), a middleware platform that integrates heterogeneous body sensors for enabling the remote monitoring of patients and improving medical diagnosis. In this paper, we present the EcoHealth architecture and a performance evaluation with an e-health application developed as proof of concept.*

1. Introdução

E-health pode ser definida como a prática de *health care* realizada com o auxílio de dispositivos eletrônicos e tecnologias de informação e comunicação (TICs), incluindo registros médicos e prescrições eletrônicos e monitoramento remoto de pacientes [Boric-Lubecke et al. 2014]. No contexto de *e-health*, pequenos sensores sem fio são colocados no corpo humano ou sob a roupa a fim de aumentar o conforto e a mobilidade do paciente que os utiliza, o qual é monitorado independentemente de sua localização e sem impacto em suas atividades cotidianas [Yuce 2013]. Com isso, é possível aumentar

a qualidade de serviços médicos oferecidos e ao mesmo tempo reduzir o seu custo, além de permitir o desenvolvimento de inúmeras aplicações, como, por exemplo, o monitoramento remoto, prevenção e tratamento de pacientes com doenças crônicas em ambientes de vida assistida (AAL – *Ambient Assisted Living*).

O contexto de *e-health* é apenas um dos domínios de aplicação que podem beneficiar-se da adoção do paradigma de Internet das Coisas (IoT – *Internet of Things*), como discutem alguns estudos recentes [Bui and Zorzi 2011, Bazzani et al. 2012, Sebestyen et al. 2014]. Na visão da IoT, qualquer dispositivo pode ser identificado, endereçado, controlado e monitorado via Internet [Atzori et al. 2010]. Nessa perspectiva, esses dispositivos inteligentes tornar-se-iam aptos a se comunicarem entre si e com outros recursos físicos e virtuais disponíveis na Internet, provendo assim diversas funcionalidades e informações acerca do ambiente no qual estão implantados. Dessa forma, em *e-health*, sensores corporais seriam objetos inteligentes capazes de proporcionar o monitoramento remoto de sinais vitais de pacientes a um baixo custo e em tempo real. Com isso, situações de emergência poderiam ser detectadas e adequadamente tratadas de maneira mais rápida, além de possibilitar diagnósticos médicos mais elaborados com base nas informações providas pelos dispositivos. Mais ainda, considerando que tais dispositivos podem comunicar-se uns com os outros, um dado sensor poderia fazer uso de informações providas por outro sensor para aumentar a qualidade das informações a serem providas a aplicações e usuários e a precisão de processos de tomada de decisão.

Apesar do imenso potencial trazido pela adoção da IoT em *e-health*, considerar sensores corporais como objetos inteligentes traz alguns desafios. O principal deles diz respeito à interoperabilidade entre tais objetos, visto que seria necessário lidar de forma idealmente transparente com uma miríade de dispositivos heterogêneos de vários fabricantes, cada um provendo uma interface de comunicação diferente e criando assim barreiras operacionais para o seu uso de forma sistêmica [Bui and Zorzi 2011]. Além disso, é necessário considerar uma grande quantidade de dados críticos a serem continuamente transmitidos pela rede, levantando questões de segurança e privacidade.

Para endereçar essas questões, plataformas de *middleware* têm surgido como uma solução promissora principalmente para prover interoperabilidade e gerenciar a crescente variedade de dispositivos associados a usuários e aplicações que fazem uso dos dados por eles providos [Bandyopadhyay et al. 2011]. Em *e-health*, uma plataforma de *middleware* tornaria possível integrar essa diversidade de sensores heterogêneos bem como disponibilizar os dados coletados via Internet. Além disso, através de interfaces de alto nível, seria possível acessar dispositivos de maneira transparente e permitir o desenvolvimento de uma gama de aplicações, reunindo e transformando informações provenientes dos dispositivos e provendo informações de valor agregado a usuários.

Nesse contexto, a **EcoHealth – Ecossistema Web de Dispositivos de *Health Care*** foi desenvolvida como uma plataforma de *middleware* para conectar médicos e pacientes utilizando sensores corporais acoplados e atuadores. O seu objetivo principal é integrar informações obtidas a partir de tais sensores heterogêneos para fins de monitoramento, processamento, visualização e armazenamento desses dados, bem como de notificação e atuação referentes a condições atuais de pacientes e seus sinais vitais. A EcoHealth caracteriza-se como uma customização, para o domínio de *e-health*, da EcoDiF (Ecossistema Web de Dispositivos Físicos) [Pires et al. 2014], uma plataforma Web genérica para IoT cujos principais objetivos são integrar dispositivos físicos heterogê-

neos e prover funcionalidades tais como visualização, processamento e armazenamento de dados oriundos desses dispositivos. A EcoHealth estende a EcoDiF em termos: (i) da implementação de um novo paradigma de comunicação que permite que a plataforma envie comandos de atuação para as plataformas de *hardware* que dão suporte aos sensores e atuadores; (ii) de um novo modelo relacional e dos elementos de código necessários para gerencia-lo, compreendendo aspectos tais como registros de pacientes, serviços de emergência, médicos, sensores e atuadores, e; (iii) de *drivers* específicos que dão suporte a sensores e atuadores relacionados ao domínio de *e-health*.

O projeto e a implementação da EcoHealth são essencialmente baseados em princípios REST (*REpresentational State Transfer*) [Fielding 2000, Webber et al. 2010] e em padrões e tecnologias Web atuais, e.g., HTTP e URIs. Uma visão geral da EcoHealth e sua arquitetura foram inicialmente introduzidas em um trabalho anterior [Maia et al. 2014]. Este artigo, por sua vez, apresenta uma versão mais avançada da arquitetura da EcoHealth e uma aplicação desenvolvida a título de prova de conceito para sua validação. Além disso, são mostrados resultados de experimentos computacionais realizados sobre um protótipo da plataforma a fim de avaliar o seu desempenho.

O resto deste artigo está organizado como segue. A Seção 2 apresenta a EcoHealth e sua arquitetura. A aplicação de *e-health* desenvolvida e os experimentos realizados são apresentados nas Seções 3 e 4. A Seção 5 discute trabalhos relacionados. A Seção 6 contém considerações finais e aponta direções para trabalhos futuros.

2. EcoHealth: Arquitetura

A EcoHealth é organizada em vários módulos que compõem sua arquitetura lógica, conforme ilustrado na Figura 1.

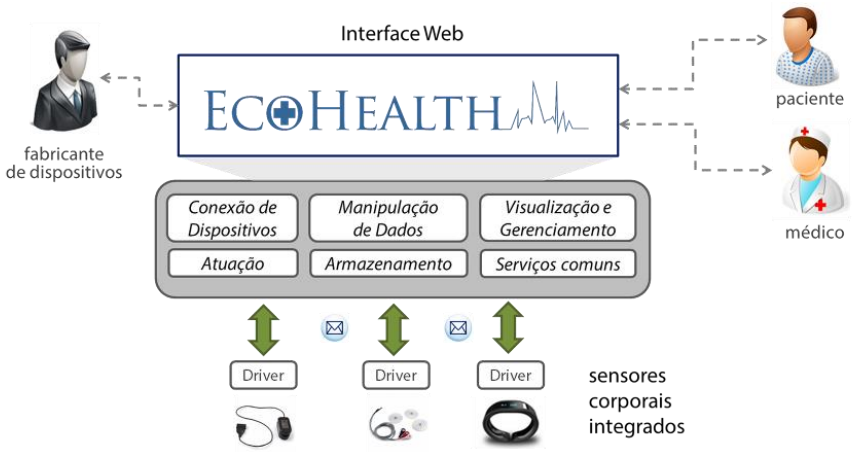


Figura 1. Arquitetura da EcoHealth.

O **Módulo de Conexão de Dispositivos** visa integrar dispositivos físicos (sensores corporais) à plataforma, utilizando, para isso, *drivers* customizados para cada tipo específico de dispositivo. Esses *drivers* desempenham um papel importante no tocante à integração de dispositivos de diferentes fabricantes uma vez que a heterogeneidade de tais dispositivos é abstraída para usuários e aplicações. Na EcoHealth há dois tipos de *drivers: ativos* e *passivos*. *Drivers* ativos obtêm informações dos dispositivos através da realização de requisições periódicas mesmo quando os valores das informações providas permanecem inalterados, enquanto *drivers* passivos aguardam por notificações pro-

venientes dos dispositivos em caso de modificações nas informações sensoriadas. Os *drivers* da EcoHealth são construídos fazendo uso de princípios REST e de padrões e protocolos Web, como HTTP e URIs. Assim, o protocolo HTTP não é usado apenas para transportar dados, mas como mecanismo padrão para interagir com os dispositivos integrados à plataforma por meio das operações definidas no protocolo (GET, POST, PUT e DELETE) e eliminando problemas de compatibilidade entre diferentes fabricantes e entre protocolos e formatos de dados [Pautasso et al. 2008]. A EcoHealth permite que a ela sejam associados diversos tipos de sensores corporais que podem ser acoplados às plataformas de *hardware* mais comumente utilizadas, tais como o Arduino[1]. Além disso, a plataforma dispõe de *drivers* para suporte a sensores como sensor de temperatura corporal, fluxo de ar, eletrocardiograma, pulso e oxigênio no sangue, posicionamento de paciente (com base em acelerômetro), e resposta galvânica da pele (suor).

Após obter dados (*feeds*) provenientes dos dispositivos integrados, os *drivers* estruturam-nos usando EEML (*Extended Environments Markup Language*)[2], uma linguagem baseada em XML para descrição de dados obtidos em um contexto específico – neste caso, sinais vitais corporais. Em seguida, tais dados estruturados são enviados para a EcoHealth por meio de requisições HTTP PUT, a fim de que sejam registrados na plataforma pelo **Módulo de Manipulação de Dados**.

No contexto de *e-health*, permitir uma comunicação direcionada aos dispositivos físicos integrados é importante principalmente em casos nos quais atuadores são utilizados para emitir sons de alerta em situações de emergência ou para administrar ações terapêuticas, por exemplo [Chen et al. 2011]. Todavia, a EcoDiF, precursor da EcoHealth, não dispõe de um canal de comunicação no sentido plataforma–dispositivos, limitando-se apenas ao envio de dados a partir dos dispositivos (por meio dos *drivers*) para registra-los na plataforma. Dessa forma, a EcoHealth adiciona essa capacidade por meio do **Módulo de Atuação**, permitindo o envio de comandos da plataforma diretamente para os dispositivos a fim de realizar alterações em suas configurações e/ou fazer com que eles desempenhem tarefas específicas. Nessa nova perspectiva, a EcoHealth (que armazena os endereços dos dispositivos ou de *gateways*, para o caso de dispositivos sem capacidades de suporte às pilhas de comunicação) realiza requisições HTTP PUT para enviar comandos de atuação para os dispositivos, comandos esses que devem ser cadastrados no momento em que os dispositivos são conectados à plataforma. Além disso, diferentemente dos *drivers* disponíveis na EcoDiF, os *drivers* de integração da EcoHealth necessitam executar um servidor Web capaz de responder tanto a requisições HTTP GET, realizadas para obter *feeds* a partir dos dispositivos, quanto a requisições HTTP PUT, realizadas para enviar comandos de atuação para os dispositivos. Ao receber uma requisição HTTP PUT, o *driver* faz o processamento desta e envia efetivamente o comando para o atuador. Se nenhum erro ocorrer, o *driver* envia uma requisição HTTP em resposta à plataforma para que ela saiba a atuação funcionou de fato. É importante destacar que os comandos de atuação que podem ser realizados dependem das capacidades de atuação providas pelos respectivos dispositivos.

Em termos práticos, a atuação é realizada por meio requisições HTTP PUT via uma URI no formato `http://<ip_dispositivo>/<atuador>`. Dessa forma, por

[1] Arduino: http://arduino.cc/
[2] Extended Environments Markup Language (EEML): http://www.eeml.org/

exemplo, a URI `http://192.168.0.1/light` poderia ser utilizada para controlar as luzes de um ambiente. O controle pode ser ainda mais preciso dependendo dos tipos de atuadores disponíveis e da implementação do *driver*. Por exemplo, poder-se-ia ter uma URI `http://192.168.0.1/light/1` para indicar o envio de mensagens para a luz `1`, ou ainda a URI `http://192.168.0.1/light/emergency`, para indicar o envio de mensagens somente para as luzes de emergência. A ideia é prover pelo menos uma URI para cada atuador disponível na plataforma de *hardware* de modo que se tal plataforma controla um dispositivo de som, a URI `http://192.168.0.1/sound` deveria ser provida pelo driver, possibilitando o seu uso pela EcoHealth. Além da URI, é necessário especificar no *payload* da requisição HTTP os parâmetros que indicam o tipo de ação a ser executada sobre o atuador especificada na URI. Por exemplo, o *payload* para ligar a luz especificada na URI `http://192.168.0.1/light` poderia ser codificado no formato JSON[3] como `{"mode":"on"}`. Naturalmente, atuadores mais complexos podem requerer uma lista de parâmetros maior para funcionarem corretamente. Ainda como exemplo, o atuador da URI `http://192.168.0.1/sound` pode requerer que sejam especificados o modo de funcionamento (rádio, USB, Bluetooth, etc.), a faixa a ser escolhida (dependendo do modo) e o volume.

O **Módulo de Visualização e Gerenciamento** provê uma interface Web que permite aos usuários (médicos, pacientes e administradores de sistema) gerenciar informações sobre pacientes, sensores corporais acoplados a eles, registros médicos, notificações, informações sobre serviços de emergência, capacidades de atuação dos dispositivos, etc. Usando essa interface, médicos podem visualizar informações atualizadas dos pacientes coletadas por sensores corporais, assim como informações históricas armazenadas na plataforma. Além disso, os médicos podem criar *triggers*, que são mecanismos de notificação baseados em eventos que os informarão acerca de condições críticas nos sinais corporais que estão sendo medidos (*feeds*). Por exemplo, considerando um *feed* associado com a pressão arterial de um dado paciente, um médico pode criar um *trigger* para notifica-lo sempre que a medida da pressão arterial for superior a 140/90 mmHg. De maneira adicional, também é possível associar *triggers* a capacidades de atuação dos dispositivos. Dessa forma, um *trigger* pode ser configurado para que um alerta sonoro seja emitido no dispositivo sempre que o nível de glicose do paciente ultrapassar um limite estabelecido. Assim, dependendo das capacidades de atuação disponíveis nos dispositivos, seria possível configurar diversos pares *triggers*/atuação que seriam capazes de prover maior qualidade de vida para pacientes de risco.

O **Módulo de Armazenamento** engloba um banco de dados relacional e um sistema de arquivos para armazenar todas as informações usados na EcoHealth, e.g., registros médicos, informações históricas provenientes dos sensores corporais, notificações, informações relacionadas aos pacientes e informações sobre serviços de emergência. É importante ressaltar que esse módulo pode fazer uso de uma infraestrutura de nuvem a fim de armazenar dados relacionais e arquivos, conferindo robustez, confiabilidade, segurança, disponibilidade e escalabilidade à plataforma.

Por fim, o **Módulo de Serviços Comuns** consiste em serviços de infraestrutura providos pela plataforma, como segurança (autenticação, confidencialidade, e integridade), suporte a transações, notificações, etc. Considerando a natureza crítica dos dados

[3] JavaScript Object Notation: http://www.json.org/

armazenados na plataforma, o esquema de autorização da plataforma garante que somente o médico autorizado pode acessar os registros de um determinado paciente e os dados providos pelos seus sensores corporais. É também essencial criptografar as informações transmitidas pelos sensores a fim de garantir confidencialidade e integridade dos dados.

Dadas as características de seus módulos, a EcoHealth prevê três perfis de usuários, a saber: (i) **fabricantes de dispositivos**, que desenvolvem *drivers* de dispositivos a fim de torna-los compatíveis com a API da EcoHealth; (ii) **médicos**, que continuamente monitoram os sinais vitais dos pacientes por meio da EcoHealth e usam as informações disponíveis na plataforma para prover melhores diagnósticos e resposta a emergências, e; (iii) **pacientes**, que proveem informações para a plataforma por meio de sensores corporais acoplados a eles.

3. Aplicação: Monitoramento Remoto de Pacientes

A fim de validar a EcoHealth, foi desenvolvida uma aplicação de *e-health* que tem como objetivo monitorar um paciente e indicar sua situação atual com base nos dados coletados por sensores nele acoplados. Para realizar medições de sinais vitais, foram utilizados três tipos de sensores, a saber: (i) um sensor de respiração, através de uma máscara fixada no nariz do paciente; (ii) um sensor de temperatura, fixado no braço do paciente, e; (iii) um sensor de eletrocardiograma (ECG) com eletrodos fixados no peito do paciente. Os dados coletados pelos sensores são enviados para a EcoHealth, que permite acompanhar as condições atuais do paciente com base em tais dados e também enviar notificações para o respectivo médico em caso de detecção de anormalidades nas medidas aferidas. A Tabela 1 apresenta os tipos de dados coletados pelos sensores corporais implantados no paciente e as condições tidas como normais e anormais para cada um.

Tabela 1. Tipos de dados coletados pelos sensores corporais e respectivas condições normais e anormais.

Sensor corporal	Dados sensoreados	Condições normais	Condição abaixo da normal	Condição acima da normal
Respiração	Ciclos de inalação/exalação em 60 segundos	Frequência eupneica (normal)	Frequência bradipneica	Frequência taquipneica (curta e acelerada)
Temperatura	Temperatura corporal	Normotermia (36-37º C)	Hipertermia (> 37º C)	Hipotermia (< 35,5º C)
ECG	Batimentos cardíacos	60-100 batimentos por minuto (bpm)	Taquicardia (> 100 bpm)	Bradicardia (< 60 bpm)

Os dados provenientes dos sensores fornecem importantes informações acerca das condições de saúde atuais do paciente. Por exemplo, o monitoramento da temperatura corporal é útil para detectar infecções e doenças, de modo que tal monitoramento constante permite melhor administrar o medicamento adequado e assim controlar o seu aumento. Por outro lado, outros sensores podem ser utilizados para confirmar alguns sintomas detectados pela temperatura ou frequência respiratória. Por exemplo, através do uso do sensor de ECG e da verificação da frequência de batimentos cardíacos, é possível identificar uma variedade de problemas de saúde que podem afetar um paciente, confirmando a sua condição clínica. A inerente heterogeneidade dos sensores que po-

dem ser usados dificulta a utilização sistêmica desses dados que, uma vez combinados, permitiriam um diagnóstico mais preciso por parte dos médicos. Nesse cenário, a EcoHealth pode ser utilizada para integrar os sensores corporais e monitorar variáveis biométricas em tempo de execução, bem como para gerar mensagens de alerta em caso de condições anormais. Para a aplicação em questão, os sensores de temperatura, respiração e ECG foram conectados a um *shield* Cooking Hacks e-Health[4] acoplado a uma placa Arduino Uno[5], combinação que permite a integração de uma variedade de sensores biométricos. Os batimentos cardíacos, a temperatura corporal e a frequência respiratória do indivíduo monitorado foram registrados na EcoHealth como *feeds*, de modo que um *driver* ativo desenvolvido para o Arduino Uno (i) coleta os dados medidos pelos sensores, (ii) estrutura-os no formato EEML, e (iii) os envia via requisições HTTP PUT para armazena-los na plataforma.

A Figura 2 mostra um trecho de representação EEML referente ao feed associado ao sensor de temperatura acoplado ao paciente. Na linguagem EEML, feeds são descritos através do elemento `environment` (linha 2), que representa os dados do ambiente que está sendo monitorado. O elemento `data` estrutura as informações aferidas pelo sensor e é composto por um conjunto de medições individuais (`datapoints`, linha 5), que informam o valor aferido pelo sensor em um determinado instante de tempo. A representação EEML exibida na Figura 2 informa que o valor de temperatura mais recente aferido pelo sensor (elemento `current_value`, linha 4) foi 36º C, o mesmo valor que foi coletado em medições anteriores (elementos `value`, linhas 6 a 10).

```
1:  <eeml xmlns="http://www.eeml.org/xsd/0.5.1" version="0.5.1">
2:     <environment>
3:        <data id="61">
4:           <current_value at="2014-11-11T17:09:24.000-02:00">36</current_value>
5:           <datapoints>
6:              <value at="2014-11-11T16:44:28.000-02:00">36</value>
7:              <value at="2014-11-11T16:44:29.000-02:00">36</value>
8:              <value at="2014-11-11T16:49:59.000-02:00">36</value>
9:              <value at="2014-11-11T16:50:02.000-02:00">36</value>
10:             <value at="2014-11-11T16:50:03.000-02:00">36</value>
11:             ...
12:          </datapoints>
13:          <unit symbol="°C" type="Grau Celsius"/>
14:       </data>
15:    </environment>
16: </eeml>
```

Figura 2. Trecho de representação EEML referente a dados do sensor de temperatura.

A Figura 3 apresenta um diagrama de sequência que ilustra as operações desde a coleta de dados provenientes de dispositivos ao registro destes na EcoHealth. Como descrito anteriormente, o *driver* coleta os dados mensurados pelos sensores, estrutura-os em EEML, e os envia à EcoHealth, operações respectivamente representadas pelas mensagens *getDataFromSensor*, *generateEEML* e *sendEEML*. Uma vez recebidos na EcoHealth, o objeto *EEMLResource*, integrante do Módulo de Manipulação de Disposi-

[4] e-Health Sensor Platform v2.0 for Arduino and Raspberry Pi: http://goo.gl/tdRDzr
[5] Arduino Uno: http://arduino.cc/en/Main/arduinoBoardUno

tivos, atualiza os dados referentes ao *environment* especificado (mensagem *updateEEML*), primeiramente validando a especificação EEML recebida (mensagem *validateEEML*) e em seguida atualizando os *datapoints* (*updateDatapoints*). Finalmente, o *EEMLResource* faz a verificação dos *triggers* previamente cadastrados que estão relacionados ao *feed* em questão (mensagem *checkTriggers*). Caso as condições dos *triggers* sejam avaliadas como verdadeiras, as notificações necessárias são enviadas (mensagem *sendNotifications*).

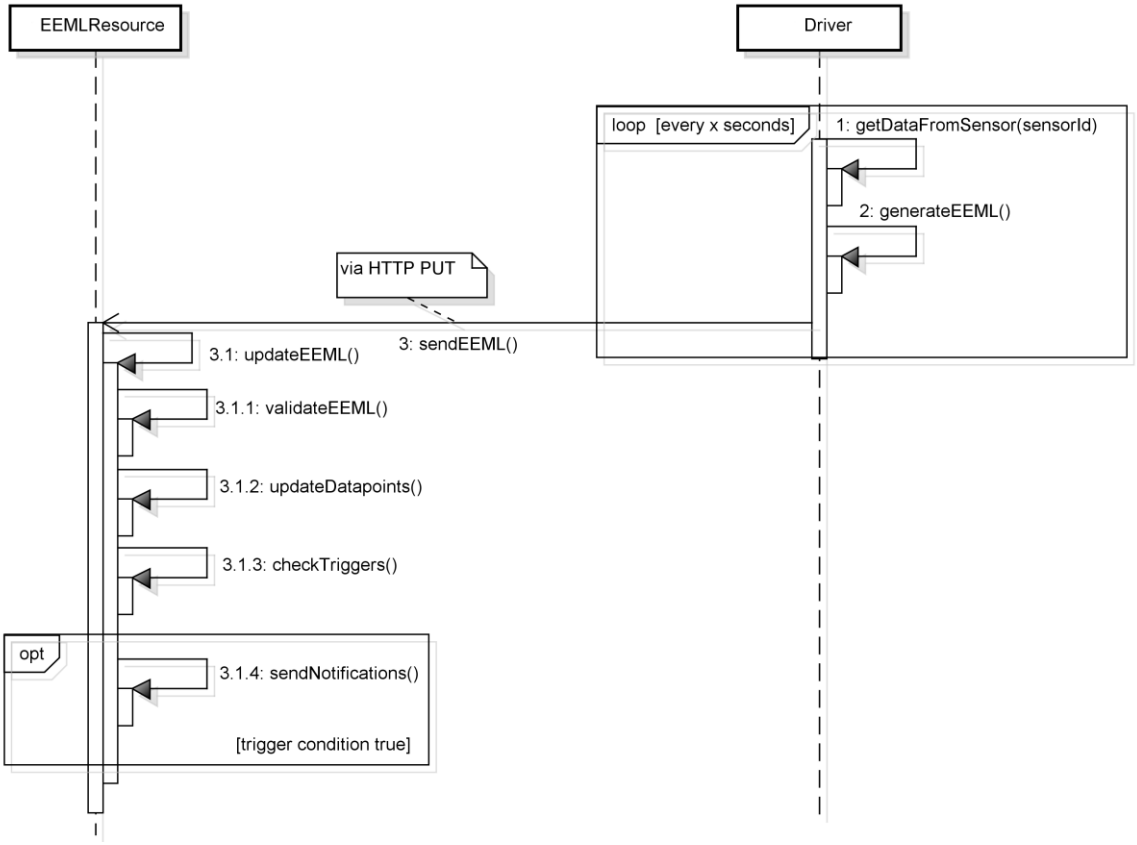


Figura 3. Diagrama de sequência de operações desde a coleta de dados provenientes de dispositivos ao registro destes na EcoHealth.

Através da EcoHealth, é possível processar os dados obtidos pelos sensores e apresenta-los a médicos e outros profissionais de saúde através de séries históricas. Por exemplo, as variações de temperatura e respiração podem ser verificadas através dos gráficos exibidos e auxiliar na detecção de variações ao longo do dia e da noite. As Figuras 4 e 5 ilustram gráficos exibidos na interface Web da EcoHealth mostrando algumas medições coletadas pelos sensores de respiração e de temperatura, respectivamente. Além disso, como anteriormente mencionado, outra funcionalidade provida pela plataforma é disparar mensagens de alerta para médicos quando as variações nos dados representam condições críticas. Dessa forma, através da EcoHealth, poder-se-ia criar um *trigger* para o *feed* associado à temperatura a fim de enviar notificações caso esta ultrapasse 37º C ou esteja abaixo de 35,5º C, que representam condições anormais de temperatura corporal (cf. Tabela 1).

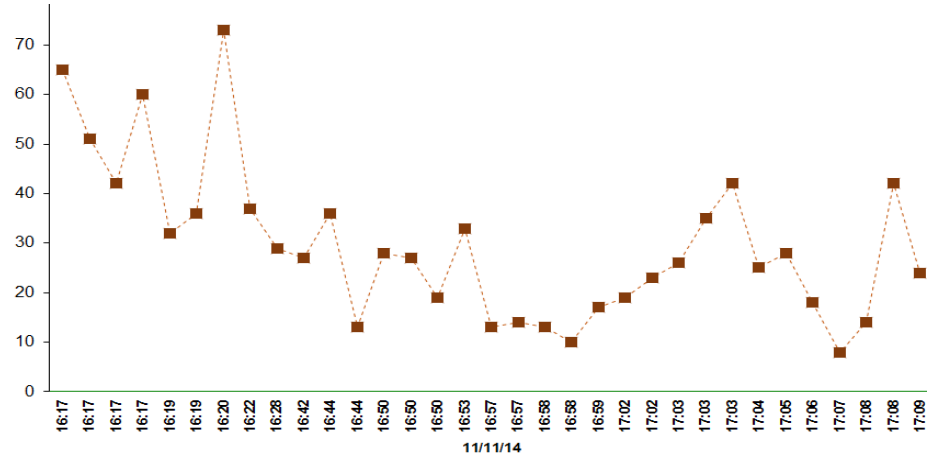


Figura 4. Histórico de monitoramento da respiração do paciente.

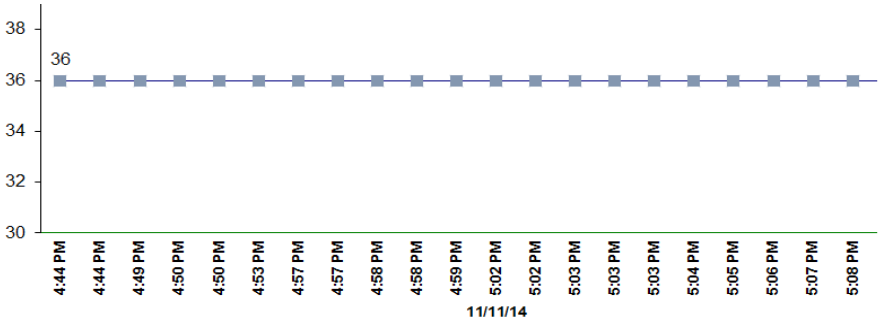


Figura 5. Histórico de monitoramento da temperatura corporal do paciente.

4. Avaliação

Em um cenário de *e-health*, pacientes são geralmente monitorados de maneira contínua, fazendo com que os dados coletados pelos sensores necessitem ser enviados com frequência à EcoHealth. Tendo isso em vista, esta seção apresenta uma avaliação de desempenho da plataforma com relação a sua capacidade de permitir uma atualização constante de *feeds* referentes aos dados provenientes de dispositivos integrados em um curto espaço de tempo. As Seções 4.1 e 4.2 detalham respectivamente o ambiente utilizado para a realização dos experimentos computacionais e a metodologia empregada, enquanto a Seção 4.3 discute os resultados obtidos.

4.1. Especificação do Ambiente de Teste

O ambiente utilizado nos experimentos computacionais realizados consistiu de um servidor Web no qual a EcoHealth foi implantada e de um computador simulando 300 dispositivos de diversos pacientes que enviam dados de temperatura corporal, frequência respiratória e batimentos cardíacos à plataforma. Os dispositivos simulados, implementados usando a linguagem Java e o componente Apache HTTP Components[6], enviam as informações para a plataforma via requisições HTTP PUT por meio de um *driver* ativo, analógo ao descrito na Seção 3. As configurações utilizadas nos experimentos foram as ilustradas na Figura 6. É importante destacar que a plataforma, durante a avaliação, re-

[6] HTTP Components: http://hc.apache.org/

cebia apenas as requisições enviadas pelo computador simulador de dispositivos, não sendo alvo, portanto, de requisições externas que poderiam interferir nos resultados a serem observados.

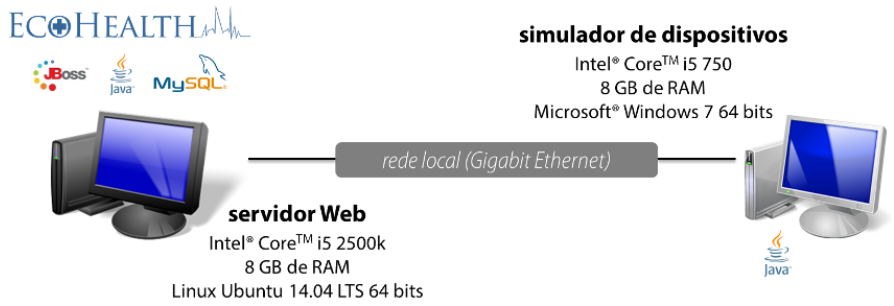


Figura 6. Ambiente utilizado nos experimentos computacionais realizados.

4.2. Metodologia

Após a implantação da EcoHealth no servidor Web, foram cadastrados 300 dispositivos e 300 *feeds*, sendo portanto cada *feed* associado a um dispositivo registrado na plataforma. É importante ressaltar que, na avaliação realizada, não se fez distinção acerca de que tipo específico de dado a ser enviado (i.e., se referente a temperatura corporal, frequência respiratória ou batimentos cardíacos), visto que a semântica dos *feeds* não promoveria alteração nos resultados e sim apenas no tamanho das mensagens enviadas.

Os experimentos realizados consistiram de cinco baterias de testes para avaliar o desempenho da plataforma. Em cada uma das baterias, 300 requisições (uma para cada *feeds*) foram enviadas 50 vezes, perfazendo assim um total de 15 mil requisições. Além disso, variou-se a frequência de envio de requisições de atualização (HTTP PUT) dos valores do *feeds* em 200, 500, 1000, 2000 e 5000 milissegundos. Como métrica, foi aferido o tempo despendido para processar cada requisição. É essencial que esses valores sejam mantidos em patamares considerados adequados para que a plataforma possa atender a todas as requisições em tempo satisfatório, de modo que mensagens de notificação possam ser enviadas em tempo hábil.

4.3. Resultados Obtidos

A Tabela 2 mostra os valores mínimo, médio e máximo, bem como o desvio padrão para o tempo de resposta em cada bateria de teste realizada, considerando diferentes intervalos entre requisições de atualização dos *feeds*. Pelos resultados obtidos, é possível perceber um decréscimo no tempo despendido para processamento das requisições à medida que o intervalo entre requisições aumenta.

Conforme mostra a Tabela 2, a partir do intervalo de 2000 ms, o servidor empregado é capaz de, em média, realizar o processamento de requisições consecutivas de um mesmo *feed* com tempo de resposta inferior ao intervalo entre requisições, de modo que a primeira requisição é completamente processada antes que a seguinte seja enviada para a plataforma. Essa é uma propriedade é importante pois, em caso de emergência, a plataforma processará rapidamente a requisição e enviará as notificações necessárias aos interessados o mais rápido possível. Ainda analisando a bateria de testes de intervalo de 2000 ms, é possível notar que, no pior caso (tempo máximo), não é possível garantir a propriedade anteriormente mencionada uma vez que o tempo

necessário para processar uma requisição foi de 3960 ms, i.e., superior ao intervalo entre requisições. Com isso, nos casos em que o tempo de processamento de requisição é superior ao intervalo entre requisições, a plataforma ainda está ocupada processando a requisição corrente quando uma nova requisição (com dados mais atuais) é recebida. Essa requisição será então tratada com atraso e eventuais notificações geradas com base na análise de tal requisição seriam enviadas também com atraso, o que poderia eventualmente colocar em risco a vida de pacientes. Essa mesma análise pode ser estendida aos demais intervalos entre requisições.

Tabela 2. Valores mínimo, médio e máximo e desvio padrão para o tempo de resposta às requisições (em milissegundos).

Intervalo entre requisições (ms)	Mínimo	Médio	Máximo	Desvio padrão
200	51	2899,88	5487	438,76
500	54	2554,89	4862	458,29
1000	37	1991,26	4550	469,91
2000	35	1216,90	3960	651,75
5000	31	305,89	2347	354,36

O tempo mínimo de processamento de requisição mostrou-se consideravelmente baixo para todas as baterias de testes, o que mostra que, em condições ideais, o processamento da requisição pode ser bastante rápido. O fato de todas as baterias de testes apresentarem um tempo mínimo semelhante é devido à ociosidade que ocorre próximo ao fim de cada bateria de teste, i.e., o número de requisições a serem processadas vai chegando ao fim. Além disso, o valor consideravelmente elevado para o desvio padrão mostra que existe uma diferença significativa no tempo de processamento entre requisições distintas em virtude das variações de carga no servidor no decorrer das baterias de teste.

Por fim, é importante ressaltar que os resultados obtidos e, consequentemente, as discussões apresentadas, estão diretamente relacionados ao ambiente de teste empregado. Em um *hardware* de melhor qualidade, a EcoHealth poderia ser capaz de processar com qualidade um intervalo menor entre requisições e/ou um maior número de *feeds* sendo atualizados constantemente.

5. Trabalhos Relacionados

Trabalhos recentes na literatura têm discutido o potencial do paradigma de IoT para o aperfeiçoamento de aplicações *e-health* bem como os desafios e oportunidades de pesquisa nesse contexto [Bui and Zorzi 2011, Boric-Lubecke et al. 2014, Sebestyen et al. 2014]. Como anteriormente mencionado, cenários de *e-health* são tipicamente caracterizados por uma alta heterogeneidade com relação aos dispositivos a serem integrados, de modo que plataformas de *middleware* para IoT seriam uma solução promissora para permitir interoperabilidade entre tais dispositivos e aplicações [Bazzani et al. 2012]. Nessa perspectiva, esta seção apresenta brevemente algumas plataformas existentes baseadas em IoT para *e-health* que se constituem como alternativas à EcoHealth.

CardioNet [Sebestyen et al. 2014] é uma plataforma Web dedicada ao tratamento de pacientes com doenças cardiovasculares utilizando o *shield* Cooking Hacks e-Health Sensor em conjunção com a plataforma Arduino Uno (de maneira similar ao cenário da aplicação apresentada na Seção 3 deste artigo) para medição de parâmetros

tais como batimentos cardíacos, pressão sanguínea, temperatura e nível de oxigenação. Além do monitoramento remoto de pacientes, o portal Web da CardioNet oferece suporte a consultas *on-line* e administração de atividades de hospitais. Todavia, essa plataforma é restrita a dispositivos e funcionalidades relacionadas a dados cardiológicos.

REACTION [Ahlsén et al. 2012] é uma plataforma de *middleware* orientada a serviços que possui por objetivo principal facilitar o desenvolvimento de aplicações em *e-health*, em particular voltadas para pacientes com diabetes. A fim de endereçar a heterogeneidade existente em termos de dispositivos, protocolos, etc., a abordagem empregada pela REACTION é de utilizar serviços Web que podem ser desenvolvidos e implantados na plataforma. Tais serviços podem ser desenvolvidos de forma independente utilizando o Open Source Medical Device Connectivity Kit (DCK), um ambiente de desenvolvimento baseado em XML e serviços Web que permite que desenvolvedores construam serviços Web de maneira compatível com a API provida pela REACTION, a fim de permitir a sua implantação na plataforma. Uma vez disponíveis, tais serviços podem ser orquestrados a fim de compor diferentes aplicações, especificadas em alto nível utilizando *workflows* que especificam a ordem de execução desses serviços.

μWoTOP (*micro Web of Things Open Platform*) [Corredor et al. 2014] é uma plataforma de *middleware* que objetiva facilitar a integração de sensores heterogêneos. Os principais elementos dessa plataforma são: (i) sensores biométricos (e.g., monitores cardíacos, acelerômetros, termômetros corporais, etc.) integrados à plataforma via *adapters*, similares aos *drivers* existentes na EcoHealth; (ii) *gateways* responsáveis por coletar dados dos sensores integrados e transmitir notificações urgentes aos interessados em caso de eventos críticos, e; (iii) aplicações clientes. Além de ser baseado em padrões Web bem estabelecidos, a μWoTOP provê um ambiente para permitir reuso e compartilhamento de recursos disponíveis na plataforma, de modo que aplicações clientes podem fazer uso de dados de sensores por meio de uma interface RESTful.

Todas essas propostas apresentadas na literatura possuem diversos objetivos em comum com a EcoHealth, principalmente com relação à integração de sensores corporais heterogêneos e à provisão, em alto nível, de informações fornecidas por tais sensores a médicos e/ou aplicações. Além disso, é possível observar certa tendência no tocante ao uso de infraestruturas de nuvem visando promover escalabilidade, disponibilidade, desempenho e utilização de recursos sob demanda. Todavia, um aspecto importante que distingue a EcoHealth de tais plataformas diz respeito à existência de mecanismos genéricos que permitam o desenvolvimento de diferentes aplicações médicas que façam uso dos dados disponíveis na plataforma. No caso da μWoTOP e da EcoHealth, tal desenvolvimento é facilitado por meio da provisão de uma interface RESTful que permite acessar os dados providos pelos dispositivos de forma transparente e ainda compatível com o paradigma de WoT. Finalmente, dentre as plataformas mencionadas nesta seção, apenas a μWoTOP e a EcoHealth oferecem mecanismos de notificação com base nas medidas aferidas pelos sensores corporais integrados.

6. Conclusão

Este artigo apresentou a EcoHealth, uma plataforma de *middleware* para IoT baseada na Web para integrar médicos, pacientes e sensores corporais heterogêneos por eles utilizados. Os principais objetivos da EcoHealth são: (i) monitorar, através dos sensores corporais, diversas variáveis referentes à saúde do ambiente; (ii) possibilitar diagnósti-

cos através do controle, visualização, processamento e armazenamento de dados em tempo real, e; (iii) possibilitar a atuação em plataformas de *hardware* a fim de prover auxílio emergencial para pacientes em risco.

O projeto da EcoHealth é baseado em várias tecnologias Web bem estabelecidas (HTTP, REST e EEML) com o intuito de padronizar e simplificar o desenvolvimento de aplicações no contexto de IoT, minimizando assim problemas de compatibilidade e de interoperabilidade entre fabricantes, protocolos proprietários e formatos de dados. Neste artigo, foi apresentada a arquitetura lógica do EcoHealth e uma aplicação no domínio de *e-health* no qual a plataforma pode ser útil. Por fim, também foi descrita uma avaliação quantitativa do desempenho da plataforma, que foi submetida a uma elevada quantidade de requisições de atualizações de *feeds* em um curto espaço de tempo. A avaliação mostrou que a plataforma pode suportar uma quantidade considerável de dispositivos físicos (e.g., sensores corporais) enviando dados com frequências adequadas para o monitoramento de sinais vitais.

Em trabalhos futuros, a EcoHealth será validada por meio de estudos de caso em diversos cenários reais nos quais pacientes e médicos estarão usando a plataforma. Além disso, pretende-se prover suporte ao desenvolvimento de diferentes aplicações sobre a EcoHealth a fim de permitir que elas tenham acesso às informações disponíveis na plataforma. Tais aplicações podem ser desenvolvidas como *Web mashups* [Guinard and Trifa 2009], uma tecnologia que tem sio considerada adequada para o paradigma de IoT e que consiste de aplicações *ad-hoc* criadas pela composição de diferentes tipos de informações provenientes de diferentes fontes.

Agradecimentos

Esse trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e pelo Instituto Nacional de Ciência e Tecnologia para Engenharia de Software (INES)[7]. Flavia C. Delicato é parcialmente financiada pelo CNPq através dos processos 310661/2012-9, 457783/2014-1 e 307378/2014-4. Thais Batista é parcialmente financiada pelo CNPq através dos processos 573964/2008-4 (INES) e 308725/2013-1. Flavia C. Delicato, Thais Batista e Paulo F. Pires são bolsistas de produtividade do CNPq.

Referências

Ahlsén, M. et al. (2012) "Service-oriented middleware architecture for mobile personal health monitoring". In: Nikita, K. S. et al., eds. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 83. Germany: Springer-Verlag Berlin Heidelberg. pp. 305–312.

Atzori, L. et al. (2010) "The Internet of Things: A survey", Computer Networks, vol. 54, no. 15, pp. 2787–2805.

Bandyopadhyay, S. et al. (2011) "Role of middleware for the Internet of Things: A study", International Journal of Computer Science & Engineering Survey, vol. 2, no. 3, pp. 94–105.

[7] http://www.ines.org.br/

Bazzani, M. et al. (2012) “Enabling the IoT paradigm in e-health solutions through the VIRTUS middleware”, Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, IEEE Computer Society, USA, pp. 1954–1959.

Boric-Lubecke, O et al. (2014) “E-healthcare: Remote monitoring, privacy, and security”, Proceedings of the 2014 IEEE MTT-S International Microwave Symposium, USA, pp. 1–3.

Bui, N., Zorzi, M. (2011) “Health care applications: A solution based on the Internet of Things”, Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies, ACM, USA.

Chen, I. M. et al. (2011) “Personalized biomedical devices & systems for healthcare applications”, Frontiers of Mechanical Engineering, vol. 6, no. 1, pp. 3–12.

Corredor, I. et al. (2014) “A lightweight Web of Things open platform to facilitate context data management and personalized healthcare services creation”, International Journal of Environmental Research and Public Health, vol. 11, pp. 4676–4713.

Delicato, F. C. et al. (2013) Middleware solutions for the Internet of Things, Springer London, United Kingdom.

Duquennoy, S. et al. (2009) “The Web of Things: Interconnecting devices with high usability and performance”, Proceedings of the 2009 International Conference on Embedded Software and Systems, IEEE, USA, pp. 323–330.

Fielding, R. (2000) Architectural styles and the design of network-based software architectures. PhD Thesis, University of California-Irvine, USA.

Guinard, D., Trifa, V. (2009) “Towards the Web of Things: Web mashups for embedded devices”, Proceedings of the 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web.

Maia, P. et al. (2014) “A Web platform for interconnecting body sensors and improving health care”, Procedia Computer Science, vol. 40, pp. 135–142.

Pautasso, C. et al. (2008) “RESTful Web services vs. “big” Web services: Making the right architectural decision”, Proceedings of the 17th International Conference on World Wide Web, ACM, USA, pp. 805–814.

Pires, P. F. et al. (2014) “A platform for integrating physical devices in the Internet of Things”, Proceedings of the 12th IEEE International Conference on Embedded and Ubiquitous Computing, IEEE, USA, pp. 234–241.

Sebestyen, G. et al. (2014) “eHealth solutions in the context of Internet of Things”, Proceedings of the 2014 International Conference on Automation, Quality and Testing, Robotics, IEEE, Romania, pp. 1–6.

Webber, J. et al. (2010) REST in practice: Hypermedia and systems architecture. USA: O’Reilly Media.

Yuce, M. R. (2013) “Recent wireless body sensors: Design and implementation”, Proceedings of the 2013 IEEE MTT-S International Microwave Workshop Series on RF and Wireless Technologies for Biomedical and Healthcare Applications, IEEE, USA, pp. 1–3.

Characterizing User Behavior on a Mobile SMS-Based Chat Service

Rafael de A. Oliveira[1], **Wladmir C. Brandão**[1], **Humberto T. Marques-Neto**[1]

[1]Instituto de Informática – Pontifícia Universidade Católica de Minas Gerais (PUC)
Belo Horizonte – MG – Brazil

rafael.oliveira.412093@sga.pucminas.br, {humberto,wladmir}@pucminas.br

Abstract. *The use of mobile instant messaging (IM) services has grown significantly last years. Usually, mobile chat services work over the Internet using cellphone carriers' resources, such as the SMS (Short Message Service) platforms. Understanding the user behavior in this environment is paramount to improve service performance and user experience. In this article, we present and discuss a characterization of the user behavior on a mobile SMS-based chat service. We describe the usage patterns of this service providing a daily perspective of user behavior. We show that a very small group of heavy users consumes a significant amount of carrier's resources. Moreover, we also present the transitions and navigation patterns of this very small group of users to understand their peculiar behavior.*

1. Introduction

Mobile instant messaging (IM) services have been outstanding as important communication tools by connecting an increasing number of persons at any time of the day at any place around the world. According to [Mander 2014], about 600 million adults are currently using IM services on their mobile devices provided by mobile applications like *Viber*, *Kik*, *WhatsApp*, *Line*, and *WeChat*. Usually, these applications work over Internet. Nevertheless, similar short message service (SMS) services based on the exchanging of short messages have been provided by cellphone companies around the world, such as Vodafone[1], Orange[2] and Safaricom[3]. Whereas the massive data volume generated by these services over networks' resources should be handled by mobile service providers, they need to understand the behavior of their users to improve user experience, performance, availability, cost, and quality of offered service.

The present article characterizes user behavior on a mobile SMS-based chat service provided by a major cellphone carrier in Brazil. Users pay a monthly flat rate to access a set of chat rooms provided by carrier. These rooms are organized by subjects to users send short messages to others with similar interest. They also can create private rooms to chat particularly with other users. In early 2014, about 335,000 messages per day were exchanged on this service. Considering that the service is not free and is based on SMS, this volume is enough expressive.

In particular, we provide an extensive analysis of the service's usage patterns considering a dataset composed by two million messages exchanged among more than 20

[1]http://www.vodafone.in
[2]http://www.orange.mu
[3]http://www.safaricom.co.ke

thousand anonymized users throughout one week on May 2014. We identified different user profiles using the number of exchanged messages, the number of user sessions, and the frequency of messages exchanging as input to *X-means* clustering algorithm. In addition, we use the same features and clustering algorithm to provide a daily perspective of user behavior, thereby minimizing the effects of data aggregation. Furthermore, we present the transitions and navigation patterns considering the usage of service's rooms of a particular profile of *Heavy Users*, a very small group of users that send many messages. Moreover, we presented their navigational behavior using Costumer Behavior Model Graphs (CBMGs) [Menascé et al. 1999].

The remaining of this article is organized as follows. Section 2 presents some related work which places our work in literature. In Section 3, we describe the dataset used to characterize user behavior on the mobile chat service. In Section 4, we present a comprehensive analysis on characterization results. Section 5 describes the usage behavior and the navigation patterns of particular user profiles. Finally, Section 6 points out the final remarks and a brief discussion on future work.

2. Related Work

There is a significant set of related works in literature towards characterizing IM services. Most of them focused on user behavior, particularly on users interactions in the workplace [Isaacs et al. 2002], message traffic and conversations [Zerfos et al. 2006], user engagement [Budak and Agrawal 2013], and service architecture [Fiadino et al. 2014]. Different from previous work in literature, we provide a characterization of a private SMS-based chat service to detect malicious or atypical user behavior.

[Xu and Wunsch 2005] show that clustering techniques has been applied in a wide variety of fields, ranging from life and medical sciences, engineering (machine learning, pattern recognition), computer sciences (web mining, spatial database analysis, data mining). In this article, we use the X-means algorithm [Hall et al. 2009], an extension to the K-means [Jain et al. 1999]. The both algorithms are commonly used in characterization works [Benevenuto et al. 2012, O'Donovan et al. 2013]. However, X-means provides improved functions, such as the automatic detection of the number of clusters to generate.

In [Lipinski-Harten and Tafarodi 2013], the authors argue that online users can act improperly since the negative impact of recrimination for inappropriate behavior is lower than in face to face communication. For example, users may not be inhibited from using offensive language or disclosure of inappropriate content, such as pornography and violence in chat rooms not suitable for such content. In this line, previous work in literature have proposed approaches to detect malicious behavior in online conversations [Frank et al. 2010, Gupta et al. 2012, Wollis 2011].

In addition to prevent malicious behavior, a major challenge for IM service providers is to improve service performance preserving user loyalty [Deng et al. 2010]. In this line, there are important aspects that must be considered, such as the size of the user neighborhood represented by the number of contacts of an user, and the degree of confidence and engagement of the user with the IM service. In [Zhou and Lu 2011], the authors argue that low cost, attractive features, and extreme competition are key factors for an user to migrate from one IM service to another.

In [Du et al. 2009], the authors suggest a model to investigate user behaviors

changing on weighted time-evolving networks, based on clique patterns and other features. Considering the user patterns, the authors detected suspicious behaviors in outliers – a particular group of users.

3. Dataset

The dataset used in our analysis contains messages exchanged on a mobile SMS-based chat service provided by a major cellphone company in Brazil[4] during the week from May 10^{th} to May 16^{th}, 2014. The dataset includes 2,348,805 messages exchanged by 21,210 users who visited 34 different categories of chat rooms. The message exchanging occurs within 95,235 different sessions created by users. For privacy, user identifications were completely anonymized. Each record of the dataset represents one message sent by an user and contains the following fields:

- *Session Identifier*: an unique identifier of one user session; a new user session is created every time user initiates a navigation over the rooms of the mobile chat; after a downtime of 30 minutes, user session is finished.
- *Sender*: an unique identifier (anonymized) of the user that sent the message.
- *Category Identifier*: an unique identifier of the chat room category.
- *Category Name*: the name (label) of the chat room category.
- *Message*: the content of the message.
- *Message Type*: an unique identifier of the message type, i.e. *Private*, *Public*, and *Room*.
- *Timestamp*: sending message date and time.

The messages exchanged by users can be (i) *Public*, i.e. messages sent and accessible to all users in the chat room, (ii) *Room* messages sent to a single user but accessible by all users in the chat room, or (iii) *Private* messages sent to a single user and only accessible by this single user (one-to-one message).

The chat rooms are classified by their respective subjects, such as entertainment, sports, and cities, and by the nature of the content of their messages, such as restricted to 18 years old or elder. The *personal* class is used to identify chat rooms created by users. For analysis, we reorganized these chat room classes in categories as follows:

- *General*: messages of sports or religions.
- *Location*: messages related to cities and regions.
- *Person*: messages in personal chat rooms.
- *Relationship*: messages about nightlife or flirting.

4. Mobile Chat Service Overview

Different from other popular IM players such as *Viber*, *Kik*, *WhatsApp*, *Line*, and *WeChat*, which provide mobile applications with rich interfaces and a sort of facilities on the screen, the chat service considered in the present work is totally SMS-based. For instance, if a user is in a chat room and want to send a message to another user in the same chat room, the sender user must send the sequence of commands "*T + destination nickname + text message*", where T is the abbreviation to *Talk*. There are a lot of another commands that vary according to the context in which the user is in the service, for example view the available categories, the rooms of a certain category, perform administrative actions such as changing the nickname among others. In addition, there is a significant user engagement, as the service has about 335,000 messages exchanged during one day.

[4]To avoid violate privacy policies, company name and dataset details will be preserved.

4.1. Messages by Categories

Figure 1 presents the message exchanging in the mobile chat service on a daily perspective. The messages are organized by chat rooms' categories. From Figure 1, we observe that the highest amount of messages exchanged in a day occurs on Wednesday, corresponding to 14,95% of all exchanged messages in the week. Additionally, the lowest amount of message exchanging in a day occurs on Sundays and Mondays.

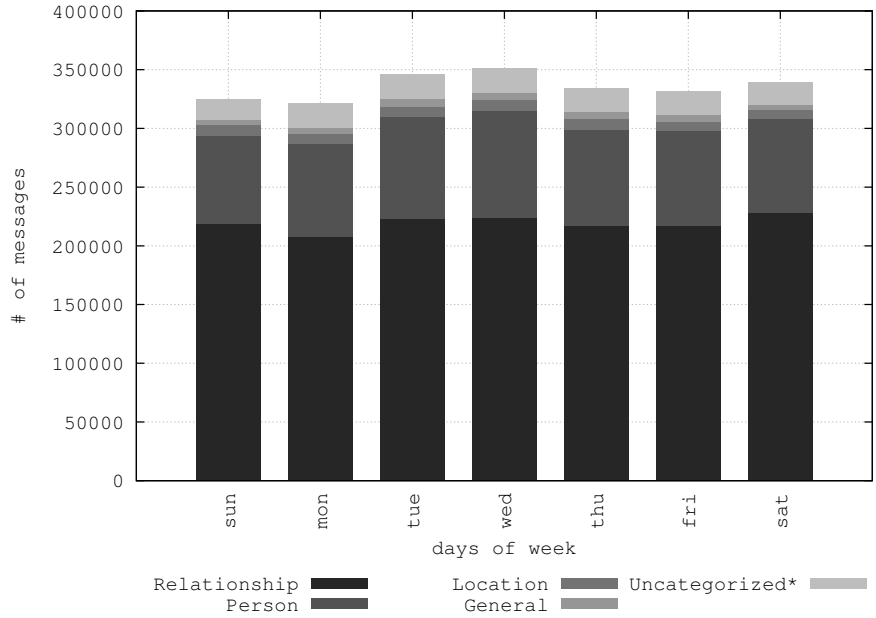


Figure 1. Messages exchanging by day and by category. *Uncategorized* messages refers to *Private* messages.

We can also observe from Figure 1 that Relationship messages correspond to 65% of all message exchanging during the week. Note that, 24% of messages are exchanged inside *"Person"* chat rooms, where users can talk about different subjects. Moreover, about 89% of all messages are exchanged in a small number of chat rooms without a specific subject.

Figure 2 presents the amount of exchanged messages over the hours of each day of the week. The darker area represents the greater amount of exchanged messages in each hour of the day. From Figure 2, we observe that highest peaks of usage occur commonly in the evenings, from 6pm to 10pm. In this time range, occurs about 36% of all message exchanging. During the afternoons, the amount of exchanged messages is also significant, corresponding to 26% of all messages. As expected, the message exchanging declines from 1am to 7am.

Nevertheless, the amount of messages exchanged per day does not vary significantly, what is very common in network traffic, but it does not occur in the SMS application. As this service creates opportunities to entertainment and social relationships, we believe the evening massive usage is related to a kind of "social need" of users. The non-occurrence of a weekly fluctuation and the high use of service in the evenings could be explained by this need, as we can observe from Figures 1 and 2.

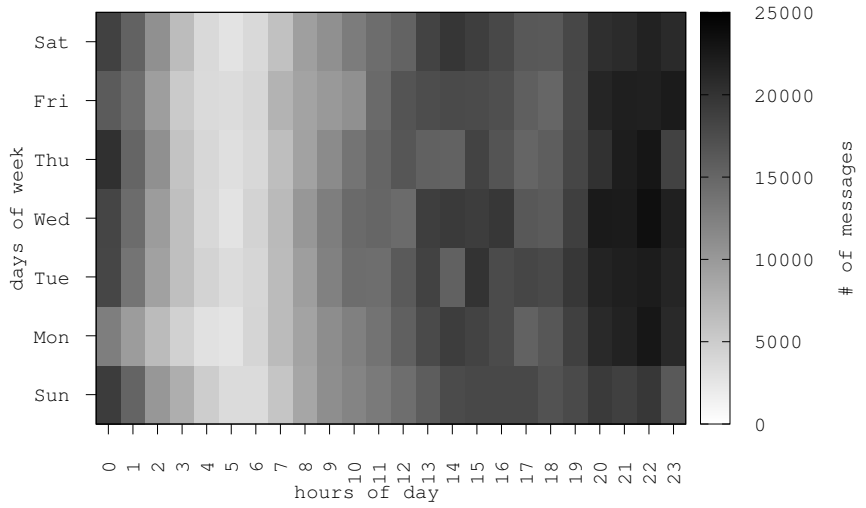


Figure 2. Message exchanging throughout the day

4.2. User Sessions and Message Types

In this section, we present two *Venn Diagrams* to represent the amount of sessions created by users and the number of messages of each category, respectively. The numbers on the labels represents the related field on the diagram. For example, from Figure 3, we observe that 45,049 user sessions contains exclusively room messages. We also observe that in 7,950 user sessions the three type of messages are present.

From Figure 3, we observe that in more than 87% of the user sessions we have exclusively *Public* and *Room* messages, suggesting a non-confidentiality pattern in the message exchanging. Moreover, almost half of user sessions are exclusively formed by *Room* messages, which suggests that users mostly communicate pairwise, but without worrying about the privacy of the communication.

Figure 4 shows that almost 77% of the messages are exchanged in non-confidential user sessions, i.e. user sessions where only *Public* or *Room* messages are exchanged. This "open communication" suggests user interest for new relationships. Additionally, more than 22% of messages are exchanged in non-exclusively confidential user sessions, while less than 1% of the messages are exchanged in private user sessions. Thus, many users build new relationships in non-confidential user sessions, and some of them intensify existing relationships in private user sessions, probably motivated by the communication context and mutual interest.

The recognition of communication context can help to characterize user behavior, since the message exchanging motivated by a specific interest follow regular patterns [Greenfield and Subrahmanyam 2003]. However, context recognition in non-confidential user sessions is a challenging problem, since many users are sending messages at the same time, frequently changing the conversation subject.

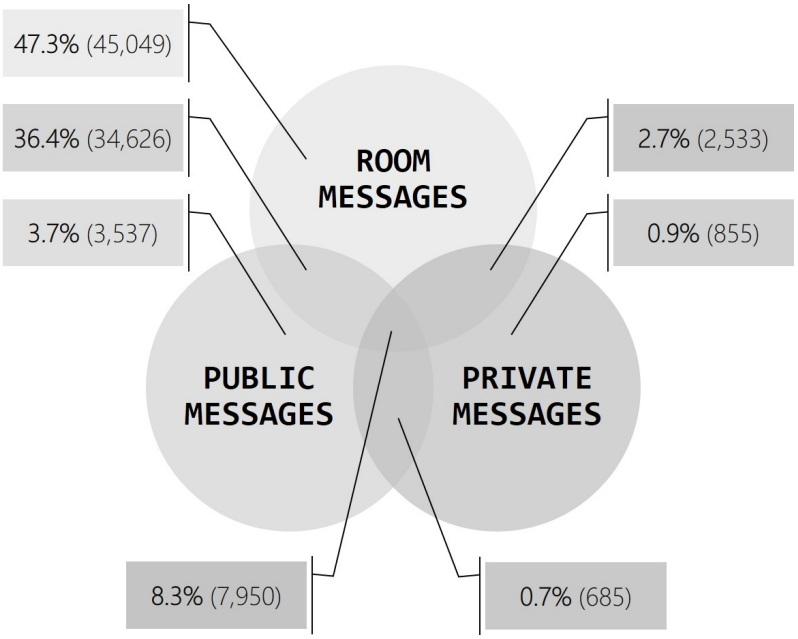


Figure 3. User sessions by message type

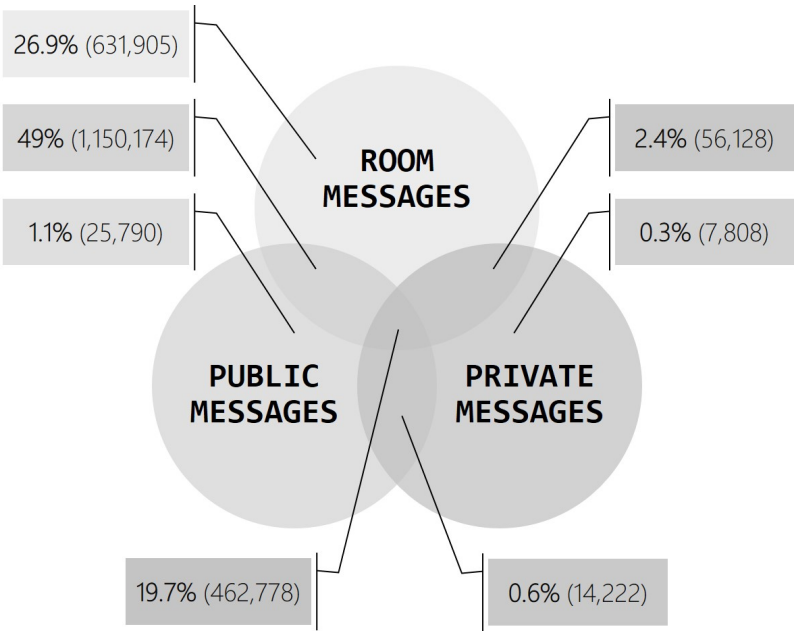


Figure 4. Messages by type on user sessions

5. User Behavior Analysis

We divide the user behavior analysis into three parts: (i) analyzing user message exchanging distribution; (ii) discovering user profiles using clustering techniques; and (ii) analyzing user transition and navigation patterns across chat rooms.

5.1. User Message Exchanging Distribution

In this section, we present the user message exchanging distribution in the mobile chat service. From Figure 5 we can observe that the user message exchanging behavior follows

a heavy-tailed distribution [Clauset et al. 2009], with a very small number of users sending the majority of the messages and the most of the users sending a very small number of messages on the chat service.

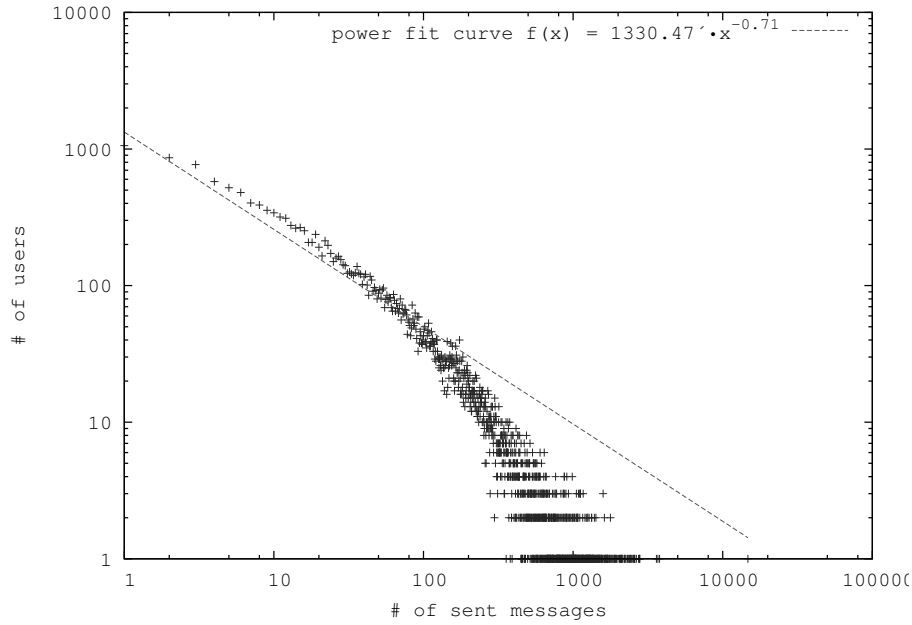


Figure 5. User message exchanging distribution.

Heavy-tailed distributions characterize an important number of behaviors from nature and human endeavor and have significant consequences for our understanding of natural and man-made phenomena. Particularly, in this article we show different user behavior on the chat service focusing our analysis on the head of the heavy-tail distribution, in a special and very small group of users which exchanges the majority of the messages.

5.2. Discovering User Profiles

In the following sections, we present a detailed characterization about user profiles who use the mobile chat service. We analyzed data in weekly and daily perspectives to understand user behavior.

5.2.1. Weekly Perspective

As aforementioned in Section 3, one user session is created every time an user initiates a navigation in the mobile chat service. Inside the session, the user exploits several chat service resources, such as listing available chat rooms by category and requesting support service. In this article, we only use the message exchanging service to discover user profiles, i.e., sets of users with similar behavior. Particularly, we consider three features about each user as input to the clustering algorithm which groups similar users:

- **Messages**: the number of exchanged messages.
- **Sessions**: the number of user sessions.
- **Frequency**: the rate of message creation per minute.

We use the X-means clustering algorithm [Pelleg et al. 2000] to discover user profiles. The X-means algorithm extends the popular K-means algorithm [Jain et al. 1999] by not only providing the clusters, but also estimating the suitable number of clusters should be created. These algorithms have been commonly used in clustering problems [Benevenuto et al. 2012, O'Donovan et al. 2013]. X-means creates clusters by minimizing the sum of the squared distances between each vector representing the averaged properties of each group and the cluster's centroid. The distance between two vectors is computed by the Euclidean distance.

In this article, we use a well known implementation of the X-means algorithm [Hall et al. 2009] setting the maximum number of clusters to 10. Table 1 shows the four clusters provided by X-means in a weekly perspective, the percentage of users in each cluster, as well as the respective features (average values) for each cluster. In addition, it presents the coefficient of variation (*(CV, i.e.* $\frac{Std.Dev.}{Average}$ *)*) for each feature to help understanding how cohesive is the cluster.

Table 1. Cluster's overview in a weekly perspective

Cluster	Users	Messages		Sessions		Frequency	
	%	Avg	CV	Avg	CV	Avg	CV
Light	65.00	33.16	1.59	1.55	0.48	0.77	9.43
Infrequent	25.00	156.08	0.94	6.26	0.34	0.59	2.89
Frequent	8.00	440.59	0.86	16.62	0.24	0.57	0.58
Heavy	2.00	934.63	0.99	36.47	0.29	0.67	0.55

The first cluster contains 65% of all users. Users in this cluster exchanged few messages, approximately 33 per user session. The average frequency of message exchanging is almost 1, which is considered a high interaction frequency. However, users in this cluster typically access the service less than twice during the week. We named this user profile as *Light Users*.

About 25% of users are in the second cluster. Users in this cluster exchanged more messages than *Light Users*, approximately 156 per user session. The average frequency of message exchanging for this cluster is slightly lower, approximately 0.6. Users in this cluster typically access the service six times during the week. We named this user profile as *Infrequent Users*.

The users in the other two clusters exchanged several messages, using the service intensively. In the third cluster we have 8% of the users. Users in this cluster exchanged several messages and access the service about 20 times during the week. Due this behavior, we named this user profile as *Frequent Users*.

Finally, in the fourth cluster we have the remaining 2% of users which exchanged a high amount of messages. They access the service about 40 times during the week. We named this user profile as *Heavy Users*. This group represents only 2% of the users but exchanged about 14% of all messages and creates about 14% of all user sessions in the service. Due to this behavior, *Heavy Users* receive further attention in our analyzes.

5.2.2. Daily Perspective

We also use the X-means clustering algorithm and the same three features described in Section 5.2.1 to analyze the usage of the mobile chat service on a daily perspective. For comparison, we set the number of clusters to four, the same number of clusters found in the weekly perspective presented in Section 5.2.1, rather than allowing X-means to automatically discover the suitable number of clusters. Figure 6 presents the proportion of users in clusters in a daily perspective.

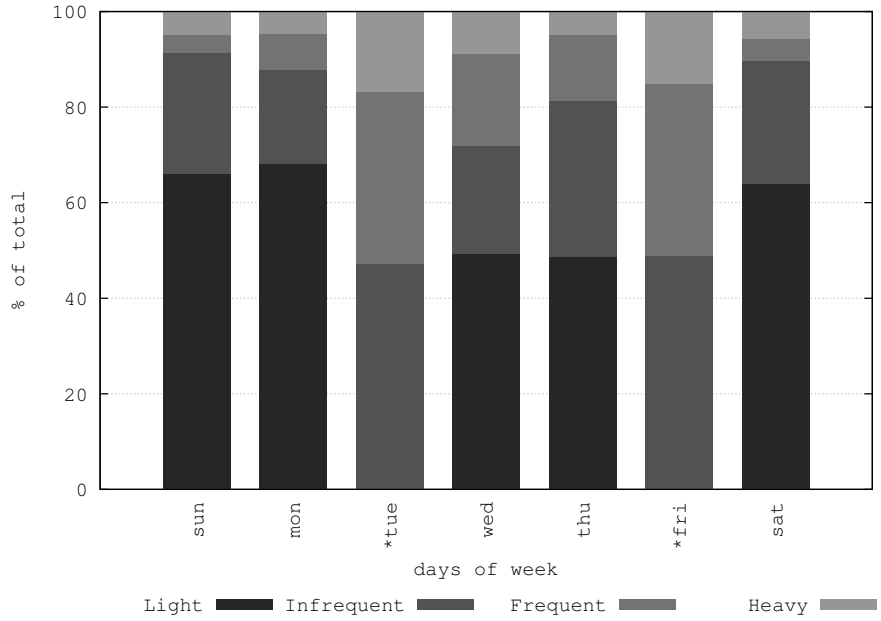


Figure 6. Proportion of users in clusters in a daily perspective.

From Figure 6, we observe that the proportion of users in clusters is similar to the weekly perspective, with a dominance of the *Light Users*, followed by *Infrequent Users*, *Frequent Users*, and *Heavy Users*. The exception occurs within two days of the week, Tuesday and Friday, when there is almost no *Light Users* using the service. In these cases, probably the *Light Users* have changed their behavior in the other days using the service more frequently.

Table 2 presents the four clusters provided by X-means in a daily perspective, as well as the respective features (average values) for each cluster. In addition, it presents the coefficient of variation (*CV*) for each feature.

Table 2. Cluster's overview in a daily perspective

Cluster	Messages		Sessions		Frequency	
	Avg	CV	Avg	CV	Avg	CV
Light	17.56	0.34	1.33	0.24	0.82	0.28
Infrequent	49.28	0.40	2.38	0.44	0.58	0.06
Frequent	112.41	0.39	4.41	0.55	0.60	0.11
Heavy	181.18	0.34	5.55	0.27	0.62	0.08

From Table 2 we observe that, similarly to the weekly perspective presented in Table 1, *Heavy Users* exchanged a high amount of messages per day, corresponding to almost 4 times more message exchanging than the *Infrequent Users* and 10 times more message exchanging than the *Light Users*, the two most representative groups. Additionally, *Heavy Users* created 3 times more user sessions than the *Infrequent Users* and 6 times more user sessions than the *Light Users*. Moreover, on a daily basis, the interaction frequency of the *Infrequent Users*, *Frequent Users*, and *Heavy Users* is almost the same. Since the average amount of exchanged messages by *Heavy Users* is significantly greater than the other groups, we conclude that *Heavy Users* use the message exchanging service for longer.

5.3. Transition and Navigation Patterns

As mentioned in Section 5.2.1, *Heavy Users* represent 2% of the users, exchanging about 14% of all messages and creating about 14% of all user sessions in the message exchanging service. In this section, we focus our analyses on *Heavy Users* investigating the user profile transition and navigation patterns of this peculiar user profile.

Particularly, to understand the user profile transitions, we identify *Heavy Users* in a day (*D*), recognizing their user profile in the day before (*D-1*). In addition, we analyse how *Heavy Users* back to the mobile chat service, recognizing their user profile in the day after (*D+1*). Table 3 presents the *Heavy Users* composition on a *D-1*/*D* perspective. The *D* parameter was defined considering users with sessions between 0:00 and 23:59. By this, we were considering a daily perspective.

Table 3. Heavy Users composition on a D-1/D perspective

Light	12.59%
Infrequent	21.91%
Frequent	20.06%
Heavy	30.99%
New Heavy Users	14.46%

From Table 3, we observe the majority of Heavy Users, almost 55%, in *D* belong to different user profile in *D-1*. In particular, almost 42% of *Heavy Users* in *D* were *Infrequent Users* or *Frequent Users* in *D-1*. Additionally, almost 13% of *Heavy Users* in *D* were *Light Users* in *D-1*. Moreover, the remaining 14% represents new *Heavy Users* that do not use the message exchanging service in*D-1*.

Table 4 presents the *Heavy Users* engagement on a *D*/*D+1* perspective. From Table 4, we observe that more than 85% of *Heavy Users* in *D* back to the message exchanging service in the next day, and about 42% of them back with the same user profile. We can conclude that *Heavy Users* tend to remain in this behavior, since almost 31% of the users in this profile were already *Heavy Users* in *D-1*.

This group of *Engaged Users* that remain *Heavy Users* over time frequently returning to the service contribute to reinforce the *Heavy Users* behavior intensively exploiting service resources.

To understand the navigation behavior of *Heavy Users*, we use a *Customer Behavior Model Graph* (CBMG), a state transition graph that has been used to describe the

Table 4. Heavy Users engagement on D/D+1 perspective

Return rate	85.18%
Light	13.21%
Infrequent	17.64%
Frequent	26.92%
Heavy	42.22%

navigation patterns of groups of users [Menascé et al. 1999]. In this graph, each edge represents a transition probability from one node to another and each node represents a possible state to reach. Figure 7 presents a CBMG of the transition behavior for user profiles in a daily perspective. In this graph, each node represents one user profile and each edge represents the transition probability between user profiles. In addition, we also represent two abstract nodes in the graph, representing the start (*entry*) and the end (*exit*) states. We also highlight the paths with the highest transition probabilities.

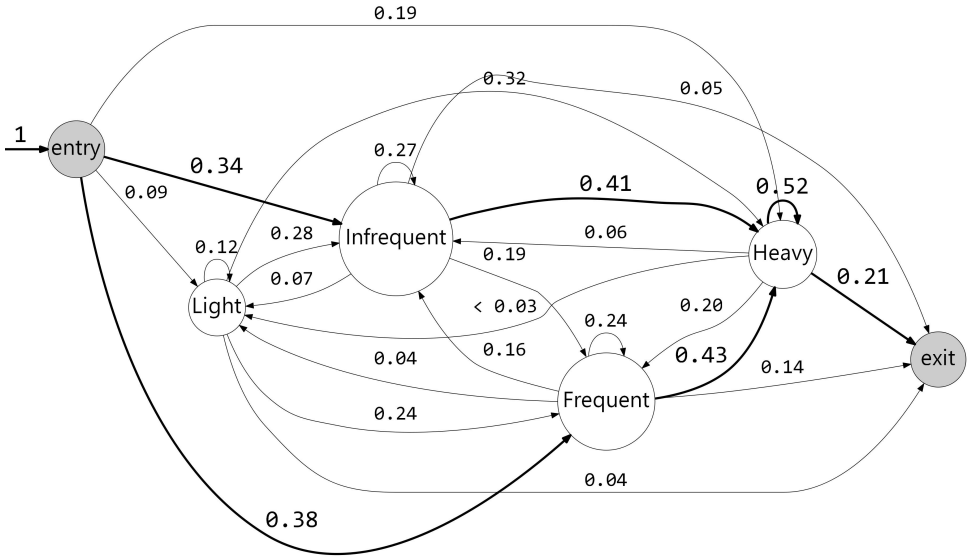


Figure 7. CBMGs for behavioral changes. The paths with the highest probability were highlighted.

From Figure 7, we observe that the *Heavy Users* change their behavior during the week. They are more likely to be initially classified as *Frequent Users*, with a probability of 0.38, followed by*Infrequent Users*, with a probability of 0.34. In both cases, users that are classified in these behavior have a high tendency to migrate to the group of *Heavy Users*, with an average probability of 0.42, remaining until the end of the period with a probability of 0.52.

Figure 8 presents a CBMG of the chat rooms exploitation by category in a daily perspective. In this graph, each node represents one chat room category and each edge represents the transition probability between chat room categories. Additionally, we also represent the abstract nodes *entry* and *exit* in the graph, and we also highlight the paths with the highest transition probabilities.

From Figure 8, we observe that *Heavy Users* usually start a session in the chat

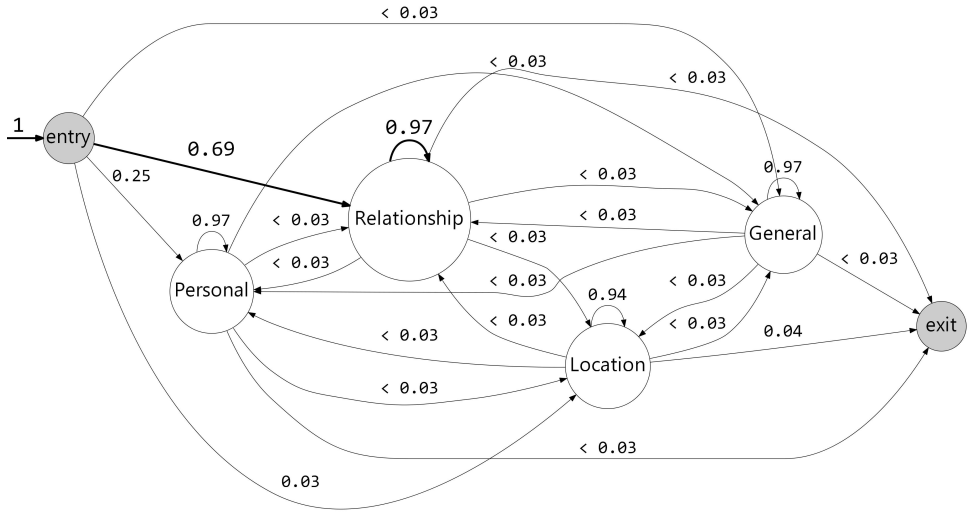


Figure 8. CBMGs for categories exploitation. The paths with the highest probability were highlighted.

through a room from the Relationship category, with a probability of 0.69. Once in a room from this category, the *Heavy Users* have an extremely high chance of staying in this type of room, with a probability of 0.97. The transitions from this state have little significant values, showing that *Heavy Users* effectively look for rooms of type Relationship.

6. Conclusions and Future Work

In this article we presented a comprehensive characterization of the user behavior on a mobile SMS-based chat service provided by a major cellphone company in Brazil. In particular, we described the usage patterns of this service using a dataset with millions of short text messages exchanged between thousands of users during a week.

In this high traffic IM service, message exchanging occurs mostly in the afternoons and evenings, in the middle of the week and inside *Relationship* chat rooms, with the majority of messages being accessible by anyone inside a chat room. Additionally, the weekly and daily perspectives of the user behavior points to the existence of four distinct groups of users: i) a large group of *Light Users* (65%) that exchanges very few messages with a very small gap between message exchanging and uses the service less than two times a week; ii) a group of *Infrequent Users* (25%) that exchanges few messages with a small gap between message exchanging and return to the service constantly; iii) a small group of *Frequent Users* (8%) that uses the service three times more frequently and exchanges more messages than *Infrequent Users*; iv) a very small group of *Heavy Users* that uses the service two times more frequently and exchanges much more messages than *Frequent Users*.

By focusing our analysis on the transition and navigation patterns of this very small group of *Heavy Users*, we show that these users tend to keep their behavior over time. In addition, they are engaged users that frequently back to the service intensively exploiting its resources. Moreover, we show that a significant part of *Infrequent Users* and *Frequent Users* change their behavior becoming *Heavy Users*. Analyzing the chat category exploitation, we show that *Heavy Users* look for Relationship chat rooms and

stay there.

The behavior patterns aforementioned about the *Heavy Users*, such as the amount of exchanged messages, the number of created user sessions, and the high service engagement, suggest be likely to find in this very small group of users those with a potential malicious behavior. Considering possible directions for future research, directly inspired by or stemming from the results of this work, we plan to investigate the message content of the *Heavy Users* to detect malicious behavior, such as defamation, pedophilia, phishing, and spamming.

We also plan to use other clustering algorithms and investigate different features, such as the distribution of messages by category, the duration of user sessions, and the message content. Another direction is to cluster user behaviors instead of users, looking for behavioral classes such as exploring and flirting. There are some techniques designed to capture roles and their dynamics, as suggested in [Fu et al. 2009, Nasraoui et al. 2008].

Moreover, we plan to further investigate transitions evolving private messages. As we observed, less than 1% of the messages are exchanged in private user sessions, suggesting that the final goal of the users is to get the contact number (e.g Whatsapp or another private way of contact) of the person, so they will be able to chat in a more friendly environment, away from any possibility of moderation. Once they do it, they will stop using the private chat (and the chat itself).

References

Benevenuto, F., Rodrigues, T., Cha, M., and Almeida, V. (2012). Characterizing user navigation and interactions in online social networks. *Information Sciences*, 195:1–24.

Budak, C. and Agrawal, R. (2013). On participation in group chats on twitter. *International World Wide Web Conference*, pages 165–175.

Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-law distributions in empirical data. *SIAM Rev.*, 51(4):661–703.

Deng, Z., Lu, Y., Wei, K. K., and Zhang, J. (2010). Understanding customer satisfaction and loyalty: An empirical study of mobile instant messages in China. *International Journal of Information Management*, 30(4):289–300.

Du, N., Faloutsos, C., Wang, B., and Akoglu, L. (2009). Large Human Communication Networks: Patterns and a Utility-Driven Generator. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Fiadino, P., Schiavone, M., and Casas, P. (2014). Vivisecting whatsapp through large-scale measurements in mobile networks. *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 133–134.

Frank, R., Westlake, B., and Bouchard, M. (2010). The structure and content of online child exploitation networks. *ACM SIGKDD Workshop on Intelligence and Security Informatics - ISI-KDD '10*, pages 1–9.

Fu, W., Song, L., and Xing, E. P. (2009). Dynamic mixed membership blockmodel for evolving networks. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1–8, New York, New York, USA. ACM Press.

Greenfield, P. M. and Subrahmanyam, K. (2003). Online discourse in a teen chatroom: New codes and new modes of coherence in a visual medium. *Journal of Applied Developmental Psychology*, 24(6):713–738.

Gupta, A., Kumaraguru, P., and Sureka, A. (2012). Characterizing Pedophile Conversations on the Internet using Online Grooming. *arXiv preprint arXiv:1208.4324*.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Isaacs, E., Kamm, C., Schiano, D. J., Walendowski, A., and Whittaker, S. (2002). Characterizing instant messaging from recorded logs. *Conference on Human Factors in Computing Systems*, pages 3–4.

Jain, A., Murty, M., and Flynn, P. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*.

Lipinski-Harten, M. and Tafarodi, R. W. (2013). Attitude moderation: A comparison of online chat and face-to-face conversation. *Computers in Human Behavior*, 29(6):2490–2493.

Mander, J. (2014). Global Web Index Trends Q3 2014. Technical report, Global Web Index.

Menascé, D. A., Almeida, V. A., Fonseca, R., and Mendes, M. A. (1999). A methodology for workload characterization of e-commerce sites. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 119–128. ACM.

Nasraoui, O., Soliman, M., Saka, E., Badia, A., and Germain, R. (2008). A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites. *Knowledge and Data Engineering*, 3.

O'Donovan, F. T., Fournelle, C., Gaffigan, S., Brdiczka, O., Shen, J., Liu, J., and Moore, K. E. (2013). Characterizing user behavior and information propagation on a social multimedia network. *IEEE International Conference on Multimedia and Expo Workshops*, pages 1–6.

Pelleg, D., Moore, A. W., et al. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734.

Wollis, M. (2011). *Online Predation: A Linguistic Analysis of Online Predator Grooming*. PhD thesis, Cornell University.

Xu, R. and Wunsch, D. (2005). Survey of Clustering Algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678.

Zerfos, P., Xiaoqiao, M., Starsky H.Y, W., Vidyut, S., and Songwu, L. (2006). A study of the short message service of a nationwide cellular network. *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 263–268.

Zhou, T. and Lu, Y. (2011). Examining mobile instant messaging user loyalty from the perspectives of network externalities and flow experience. *Computers in Human Behavior*, 27(2):883–889.

Using SMS to Transfer Small Data Packets During Periods of High Workload on Mobile Data Networks

Hayala N. Curto[1], Josemar Caetano[1], Jussara Almeida[2], Artur Ziviani[3], Carlos Henrique S. Malab[4], Humberto T. Marques-Neto[1]

[1]Department of Computer Science
Pontifical Catholic University of Minas Gerais (PUC Minas)
Belo Horizonte – MG – Brazil – 30.535-901

[2]Department of Computer Science
Federal University of Minas Gerais (UFMG)
Belo Horizonte, MG – 31.270-010

[3]National Laboratory for Scientific Computing
(LNCC/MCTI) – Petropolis, RJ – Brazil – 25.651-075

[4]Oi Telecom
Rio de Janeiro, RJ 20.230-070

{hayala.curto,josemar.caetano}@sga.pucminas.br, jussara@dcc.ufmg.br,

ziviani@lncc.br, malab@oi.net.br, humberto@pucminas.br

Abstract. *This paper proposes an SMS-based platform for transferring small data packets, allowing mobile applications to operate even in situations in which the data traffic over mobile data networks is unavailable or under congestion. We first conduct a workload characterization of a Brazilian mobile network during recent large-scale events to verify calls and SMS usage patterns at such events. This characterization pointed out that voice calls and SMS work relatively well in these large-scale events, despite frequent reports from users about the data network congestion. As both transmission and reception of SMSes tend to remain in operation even during network overload, we propose in this paper the usage of SMSes to transfer small data packets during such periods. Thus we will keep key mobile applications transferring data even during periods of high workload or unavailability of the mobile network. We believe that this work could improve users' experience during large scale events and similar situations where mobile data networks are under congestion or unavailable.*

1. Introduction

The usage of mobile applications during large-scale events, such as concerts, conferences, or sport matches, has increased considerably in recent years. Typical attendees look for the most enriching experience using the applications running on their mobile devices. However, often the mobile network in the surroundings of these events can be overloaded, it may frustate a large number of people. On the other hand, despite the overload of data network, mobile operators can still keep the network calls and SMS traffic operating with certain availability.

We note however that, during periods of network congestion, SMS traffic tends to be the last one to become unavailable. Thus, in order to improve user experience, in this paper we present a new platform with an Application Programming Interface (API) to support data traffic through SMS. This platform could improve the value of mobile application for users. Towards designing such platform, we first characterize the workload of a major mobile phone network on different days when major events (notably, major soccer matches) took place in Rio de Janeiro, Brazil. We analyze the usage behavior of three profiles of attendees of two major matches of the FIFA 2014 World Cup, including the final between Germany and Argentina on July 13, 2014. We compare these profiles with similar results computed on a dataset of calls and messages collected on the day of the final match of 2013 Brazilian Cup, which happened at the same stadium. Each profile consists of: (i) attendees who made cellphone calls and sent SMS messages, (ii) attendees who only made calls, and (iii) attendees who only sent messages.

Users in each profile tend to have different behaviors in different periods of the event. Our characterization revealed that voice calls and SMS are still in use in these large-scale events, despite frequent reports from users about the unavailability of the data network. There is always a physical limit of the available network at large-scale events. Therefore, mobile operators typically also use a Wi-Fi network in the most relevant events [Estadao 2013].

This characterization was done with anonymized datasets of cellphone calls and SMS messages, provided by a major Brazilian cellphone carrier. The dataset does not include any trace from data network. We use the position of cellular operator's base stations to get approximate location of the user. Based on the results from this characterization, we propose a platform that uses a sequence of SMSes to exchange data between any Android mobile application and web services. Allow applications to transfer small data packets is the main contribution of this work.

Moreover, Brazilian mobile operators recently announced their intent to work on changing their mobile data contracts in order to consider completely blocking the user data access (rather than only slowing down the transmission rate) once the monthly data amount contracted by user is ended. On the other hand, the offer of unlimited SMS messages is growing as marketing actions of these operators, thus motivating the design of solutions, such as our proposed platform, which can exploit them to improve user experiences with their mobile devices.

The paper is organized as follows. Section 2 discusses the related work. Section 3 presents the methodology to characterize voice calls and SMS in large-scale events. The characterization results are shown in Section 4. Section 5 introduces the proposed SMS-based platform for transferring small data packages. Section 6 presents the usage of this proposed platform. Final remarks are discussed in Section 7.

2. Related Work

[Oliver 2010] and [Chen et al. 2009] characterize SMS user behavior and present some advantages of texting as a communication tool. One of them is the worldwide availability of this technology. The SMS is well-known and well-used in both developed and developing countries, and uses a communication standard with penetration in more than 220 countries. The asynchronous nature of SMS is another positive aspect, as well as its low

cost when compared with other forms of human communication [Fjeldsoe et al. 2009]. [Oliver 2010] also build an efficient and reliable data transport protocol on top of the Short Message Service (SMS). Our SMS-based platform uses and extends this work, including a mobile phone as a gateway to allow data traffic in SMS.

The massive adoption of mobile devices has motivated researchers of different areas to study economic and behavioral aspects related to the use of such devices [Eagle and Pentland 2006, Eagle and Pentland 2009], as well as human mobility patterns [Lee et al. 2009, Candia et al. 2008, Gonzalez et al. 2008, Soper 2012, Becker et al. 2013] are inferred from it. For instance, [Ratti et al. 2006] reviewed the potential of location-based services (LBS), using mobile phones, to the urban planning community, with the developing of a graphic representation of the intensity of urban activities and their evolution through space and time.

In [Xavier et al. 2013], authors presented a methodology to analyze the workload dynamics of a cellular network during large-scale events with a specific focus on soccer matches. We extend his work, exploiting also SMS messages, in addition to cellphone calls, and by identifying usage profiles. In Section 3, we detail these new features.

The SMS usage as an alternative in environments with low availability of network resources was also proposed by [Veeraraghavan et al. 2007] and [Reda et al. 2010]. In the first paper [Veeraraghavan et al. 2007], authors presented a low-cost solution for small amounts of important information traffic, replacing PC-based systems by phone-based systems. In the second one [Reda et al. 2010], authors proposed a framework for distributing individualized personal data in environments with low availability of network resources. Experimental results showed significant time saving for the end user, through prefetching scheduled personal data. The 3G networks unavaiability was described by [Zhuo et al. 2011]. The work of [Cheung and Huang 2013] proposes the usage of Wi-Fi networks to help the mobile operator to obtain immediate capacity relief when facing the explosive growth of mobile data traffic in 3G networks. A preliminary version of this work was recently published as a 2-page extended abstract [Curto et al. 2014].

3. Methodology to characterize voice calls and SMS in large-scale events

In [Xavier et al. 2013], the authors proposed a methodology to characterize voice calls during large-scale events that aims at improving the planning and management of the cellphone network resources, specially in order to accommodate the large demands that arise during large-scale events. One of its key step is the identification of cellphone users involved with a large-scale event, which enabled the characterization of workload imposed on the carrier's infrastructure. In that paper, the dataset used to validate the proposed methodology included only voice calls made by cellphone users of a major Brazilian carrier, who are completely anonymized with an unique identification. That dataset has also information about the carrier's base stations (i.e., the geographical locations) in which each voice call started and finished. In the present work, we extend this methodology in order to improve the analyses made in [Xavier et al. 2013] by also exploiting data about SMS messages (in addition to phone calls) and by analyzing the behavior of three usage profiles. In this section, we detail these new features added to the previous methodology.

As in prior work, the main assumption behind the proposed methodology is that by identifying people who made calls and sent SMS messages through the base stations

covering the region of the event location around the time the event happened, we can focus on the group of users involved directly or indirectly with the analyzed event, here referred to *attendees*[1]. Our main goal is also to identify those attendees and understand their behavior, in terms of network usage, before and after the event. To this end, we also use completely anonymized datasets with cellphone calls and SMS messages collected at a carrier infrastructure (i.e., at its base stations) that cover both the location of the selected events and its main access routes. The proposed methodology is divided into five main steps, which are described in the following sections.

3.1. Step 1 – Defining User Profiles

Different users may exhibit different profiles in terms of the types of mobile services they most often use to communicate during large-scale events. Since different services impose different loads on the carrier's infrastructure, we choose to add an initial step in the methodology which consists of defining the user profiles.

Considering the two main types of mobile services available (voice calls and SMS messages), we envision three distinct user profiles. One profile consists of users who only make use of voice calls for communication during large-scale events. Another profile is composed by users who use only SMS messages to communicate. There is a third group of users who make phone calls and also send SMS. Such user profiles are considered in all of the following stages of the methodology.

3.2. Step 2 – Defining Geographical and Temporal Constraints

The second step of the methodology consists in defining a geographical region and the time window that delimits, spatially and temporally, the occurrence of the large-scale event. This definition must take into account the event nature, the expected audience size, and the event's venue. Once the region of interest is defined, we identify the carrier's base stations that cover this particular region. The analyses should then focus on cellphone calls and SMS messages being sent through the mobile phone network via the identified base stations.

Since it is necessary to focus on the network usage during a period of time around the event's timeframe, we also adopt the timeline notation, shown in Figure 1 for the case of a soccer match, which was proposed first in [Xavier et al. 2013]. The period marked as event corresponds to the duration of the event itself. The other periods are discussed below.

3.3. Step 3 – Identifying Attendees

In the third step, we identify users who made a cellphone call or sent at least one SMS message in one of the selected carrier's base stations at a period that includes the event. The pre_{match} and $post_{match}$ periods (see Figure 1) are periods respectively just before and just after the beginning and after the end of the event. As during the event the attendees normally pay attention to what is happening, it may cause them to not make many phone calls or send SMS messages during the event. In contrast, we expect that people communicate with each other just before or after the event. Thus, in order to identify users

[1]This is an approximation, as some of those users might not have actually attended the event (i.e., event worker or passer-by).

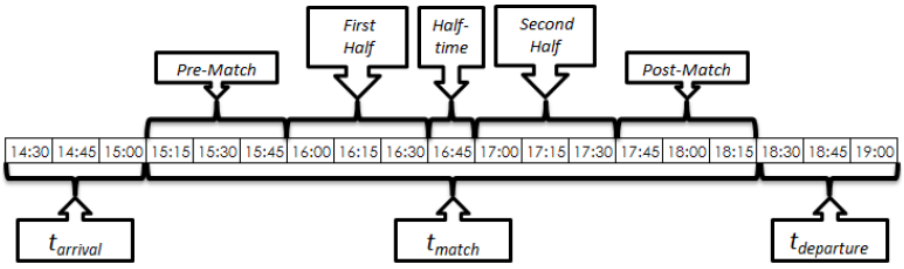


Figure 1. Timeline for workload analysis during an event

that were attending the event, here called *attendees*, we consider the total period from the beginning of pre_{match} until the end of $post_{match}$, here called t_{match}. Users who made at least one cellphone call or sent at least one SMS message during t_{match} are considered *attendees*, i.e., are a target in the conducted analyses.

3.4. Step 4 – Identifying the Movement of Attendees

In this step of the methodology, we identify the subset of attendees who also made cellphone calls or sent SMS messages before or after the soccer match, i.e. before or after the t_{match} period. This identification allows us to track the movement of these users before and after the match. To track these attendees we define two other periods: $t_{arrival}$ and $t_{departure}$ during which people are moving towards the region where the event is happening and leaving it to go somewhere else, respectively. Using the geographical location of the base stations where the voice calls were made or where the SMS messages were sent, it is possible to determine where the attendees came from and where they went to after the event. It is therefore possible to study the workload dynamics in the base stations located on the main access routes to the region where the large-scale event takes place.

For large-scale events, such as the FIFA World Cup and the Olympic Games, these main access ways possibly connect stadiums to hotel regions, to restaurants, and to airports. The identification of these routes can be exploited by many services, such as advertising or public information services.

3.5. Step 5 – Analyzing the Workload

The analysis of the workload imposed by attendees on base stations located within the region of the event, identified according to the previous four steps, focuses on the following aspects:

1. Number of distinct users and distinct attendees
2. Average number of phone calls per distinct user and per distinct attendees
3. Average number of SMS messages per distinct user and per distinct attendees
4. Average time interval between calls and SMS messages, per user and per attendees
5. IAT (Inter Arrival Time) of messages and calls at each base station.

The analysis of time intervals between successive transmissions (IATs) has great value to investigate how the phone call throughput and how transmission of SMS messages occurred. In turn, this analysis is also very important for carriers because an overload on a base station may bring degradation of service quality and consequently an interruption of service (e.g., blocking of phone calls), which could incur financial losses and

negative marketing. The analysis of SMS data, including total messages sent by users and the time intervals between transmissions by user and at each base station, along with the definition of usage profiles, can be cited as one key contributions of this work in comparison with [Xavier et al. 2013]. The mobile operator does not provide a log of drop calls. Therefore, we are unable to evaluate the system capacity.

The aforementioned aspects are analyzed not only as aggregate measures computed over the entire t_{match} period, but also at a finer granularity, measured for configurable successive time intervals of 5, 15, 30, 45, and 60 minutes during that period. Thus, workload variation during these intervals may be analyzed. To analyze the workload dynamics in the base stations located near the main access ways to the event, we used heat maps to represent the utilization rates of each base station by mobile phone services and we used these maps to represent the user movement to event venues.

The maps were generated using a function from GoogleMaps Javascript API V3, which takes latitude, longitude, and the communication calls for every analyzed base station as inputs. The region in which a base station is located is represented by a color, indicating the number of calls that went through the base station during the analyzed time span, according to the following color pattern: the darker the color, the larger the number of calls. Thus, it is possible to identify the main access ways to the event analyzing the generated maps for the $t_{arrival}$, t_{match} and $t_{departure}$ intervals.

4. Results

This section shows characterization results and is organized as follows. Subsection 4.1 describes how we applied the proposed methodology in our datasets. Subsection 4.2 characterizes user distribution and Subsection 4.3 presents communication events using timeline notation.

4.1. Applying the Proposed Methodology

According to the methodology described in Section 3, the first step is to define the user's profiles. As we only have data about phone calls and SMS messages, we defined one profile for each case (calls and SMS) and a third profile which combined both of them. The second step consists in defining the event region, and consequently the carrier's base stations that cover this region, as well as defining the time span for the analysis. For the selected soccer matches, the event region was defined as the geographical area covering a radius of 1,500 meters around the Maracanã stadium, the stadium where the matchs took place. In this case, six base stations located within this area were selected for the analysis.

Regarding the time span considered in the analysis, the total time window was the interval between 2:30 PM and 7:15 PM for the 2014 FIFA World Cup matches and the interval between 6:00 PM and 00:00 AM for the Final Match of 2013 Brazilian Cup. The total event duration, including the half-time break, is 105 minutes. We set $t_{arrival}$ and pre_{match} equal to 45 minutes each, whereas a value of 45 minutes was chosen for $t_{departure}$ and $post_{match}$ periods.

4.2. Characterizing the User Profiles

Our dataset was built from anonymized data collected by a major carrier of Brazilian cell in three important matches at the Maracana stadium. The data from Brazilian Cup final

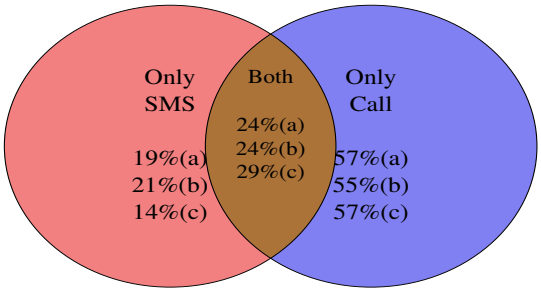


Figure 2. Fraction of identified attendees per profile.

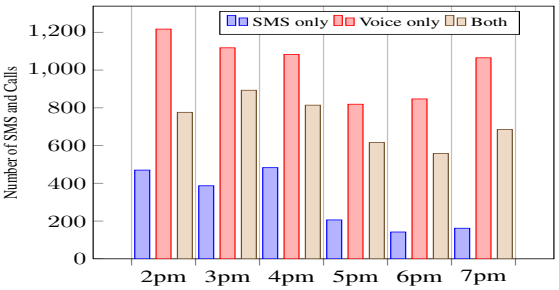


Figure 3. Profiles Timeline - Germany x Argentina

Table 1. User characterization

Profiles			SMS only		Voice only		Both	
Event	Data	Attendance	Users	% Attendees	Users	% Attendees	Users	% Attendees
Flamengo X Atlético-PR	27/Nov/13	57.991	1.428	59,17	5.830	43,57	1.362	80,10
Colombia X Uruguay	18/Jun/14	73.804	983	67,34	2.568	67,68	1.001	75,92
Germany X Argentina	13/Jul/14	74.738	1.220	34,92	3.470	48,10	1.313	64,58

match, Flamengo versus Atletico-PR was colected in November, 27, 2013. The data from World Cup Final Match, Germany versus Argentina, was colected in July, 13, 2014. We also used data from a match between Colombia versus Uruguay at World Cup, colected in June, 18, 2014. As defined, attendees are users who made at least one phone call or sent at least one SMS message during t_{match}. Figure 2 presents the percentages of identified attendees per usage profile, at each match: FLAxATL(a), COLxURU(b) and GERxARG(c). The data network characterization is out of the scope of our work because no dataset is available.

Table 1 presents the number of users and the percentage of attendees in each profile, for the three analyzed soccer matches. We note that the percentages of attendees, in SMS only and Both profiles, are smaller in the World Cup Final Match (GERxARG) than in the Brazilian Cup Final Match (FLAxATL), which indicates a more prevalent presence of users who made phone calls during the entire analyzed time span but not during the t_{match} interval. Such difference occurs in spite of a much larger attendance rate in the former. This can be explained by higher importance and the larger presence of foreigners in matches of the 2014 World Cup. According to FIFA, only 12,984 tickets were purchased by Brazilian residents (around 16% of the stadium's capacity).

4.3. Communication Events

This Section analyzes the number of phone calls and SMS messages sent by attendees during the t_{match} period, focusing on the base stations located within the selected region. Notice that, in Figure 3, we show how the numbers of communication events evolve over time (according to the identified profiles) in intervals of 60 minutes, for the final match between Germany and Argentina. The event took place between 4 and 6 PM on July 13, 2014.

Table 2 show the results for each of the proposed time intervals in the timeline ($t_{arrival}$, pre_{match}, $match$, $post_{match}$, and $t_{departure}$). Overall, as we might expect, the results show that there is a decrease in the number of phone calls and SMS messages

Figure 4. Heat Map - Voice

Figure 5. Heat Map - SMS

Figure 6. Heat Map - SMS and Voice

during the match. The amounts of these events during the $t_{arrival}$ and pre_{match} periods are similar to those of $post_{match}$ and $t_{departure}$ periods.

Table 2. User activity in the timeline – GER x ARG

Profiles	SMS only		Voice only		Both	
Timeline item	SMS	Users	Calls	Users	SMS and Calls	Users
t-arrival	876	423	2.422	1451	1.724	734
pre-match	464	235	1.004	666	760	420
first-half	145	76	440	318	400	238
halftime	61	36	379	294	215	150
second-half	100	51	567	388	391	220
pos-match	102	64	662	473	431	264
t-departure	996	477	2.571	1.471	2.104	794

We also used heat maps to analyze the mobility of attendees at each usage profile, and thus infer the most used routes to get to the event region. Figure 4 show the heat maps, considering all periods from timeline, of "Voice only" profile. In Figure 5, the heat map for the "SMS only" profile, also from all periods from timeline, is shown. Figure 6 exhibit the heat maps for the profile representing a combination of both technologies. We can see that the presence of voice calls is important for mobility studies , once we detected a small number of users at the "SMS only" profile who has mobility during events in timeline. At the match between Germany and Argentina, we observe many attendees coming from Santos Dumont Airport, specially if voice calls are used in order to infer the route.

This characterization pointed out that voice calls and SMS work relatively well in these large-scale events. As both transmission and reception of SMSes tend to remain in operation even during network overload, we propose in next Section an SMS-based platform for transferring small data packets when the data traffic over mobile data networks is unavailable or under congestion.

5. An SMS-based Platform for Transferring Small Data Packages

The Short Message Service (SMS) works in all cellphones. It allows users to send and receive short messages containing up to 160 characters. It can be used for exchanging messages between two people or to deliver the same message to a large number of people. We note that the price of this service is typically fairly low and its availability is relatively high. Indeed, the Brazilian mobile operators are including unlimited SMS plans, despite recent changes in their data plans, which impose restrictions for users to navigate at low speeds. User must purchase an upgrade of their contract or will continue using only voice calls or SMS.

Our SMS-based platform, introduced in this section, will allow the transfer of small data packets also in these situations. In the following, we first present an overview

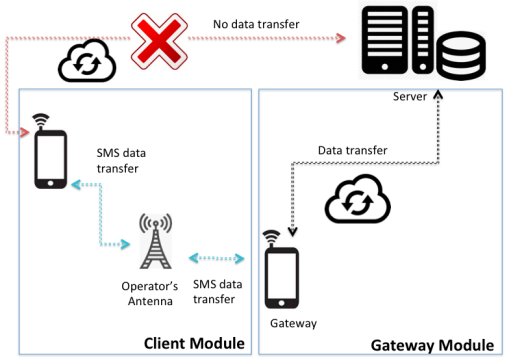


Figure 7. Architecture

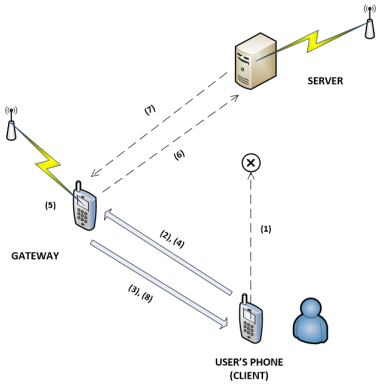


Figure 8. Operation Diagram

of our platform (Section 5.1) and then introduce its general aspects related to its implementation (Section 5.2).

5.1. Description

The main goal of the proposed platform is to keep the traffic of small data packets active even when data networks are not available or are under congestion. Thus, applications that depend on the exchange of such small data packets, running on top of our platform, would continue working. Even if the traffic resources on the data network are scalable, we know they are finite. Thus, the platform provides another communication channel, that can take care of some requests.

The trigger to use our SMS platform must be defined by these application developers. We define as *small*, packets containing up to 5 KBytes. Our platform can transmit this amount of data using approximately just 50 SMSes.

We design the platform carefully to decrease the amount of work needed to adapt existing applications. A common way to transmit data over the Internet is through mecanisms that use POST and GET calls over HTTP, as REST calls. These calls usually send and receive data using a JSON format. We maintain this format of sending and receiving data in our platform. Therefore, an application developer could test if Internet is available. If not, the developer could call our platform just as it would do for common transmission methods.

A gateway acts as a proxy between the device, which cannot access data network, and a service provider. Communication between the device and the gateway is made through a set of SMS. Communication between the gateway and the service server is made through HTTP over a Wi-fi connection. We choose to design the gateway for execution on any Android mobile device, thus allowing the use of entire platform by any individual and removing the need of third party software or hardware. Any kind of application that uses small data packets, such as instant message or news apps could use our approach.

5.2. Implementation

The platform for small data packet transfer was developed in Java for Android platform and is divided into two modules, as shown in Figure 7. The first one, Client Module is a set of methods that allow developers to perform the data transfer via SMS in a totally

transparent way, just like they would usually do for transferring data to an specific URL. The main difference in this case is the presence of a cell phone number that will act as a gateway.

The second module, or Gateway Module, is also a java application that must run on a mobile device with access to a mobile phone network to receive SMS. This device must also be connected to a wireless network, in order to transfer the information to the server addressed by the developer of the call procedures in Client Module. When the data network is unavailable, the applications that are running on the device of the user can use our platform to continue communicating with web servers. Figure 8 depicts a workflow illustrating how our platform operates. Since we are proposing a generic platform, i.e. it will be useful for any application that requires an Internet connection, we refer to a requesting application as client. The dotted arrows represent data traffic using HTTP protocols, and the solid arrows represent data traffic using SMS.

Figure 8 shows the eight steps of the operation of the proposed platform. At step (1), the client attempts to communicate with the remote server, but is not able to establish the connection. At this moment, the client asks the user for permission to continue the interaction through SMS. With permission granted, in step (2) the client calls our platform, which in turn, sends an SMS to the gateway and the connection is established between them. In (3), the gateway sends an SMS back to the client as an acknowledgement. Once the connection is established, the client uses the sendAndReceive() method whose parameter and return type are JSON objects. Thus, the platform splits the JSON object into multiple smaller packets and sends them in multiple SMS to the gateway.

In step (5), the gateway rebuilds the JSON object received via SMS. After rebuilding the JSON object, the gateway sends it through a POST to the desired Server in step (6). This step is similar to step (1), however, the gateway is using a network connection available. In step (7), the server returns the result of the POST to the gateway. In step (8), the gateway relays the POST to the platform via a sequence of SMS. The first SMS contains the result of the request and the amount i SMSes it should receive until the result of the POST can be rebuilt (like GET request). After the reconstruction, the JSON object is returned to the client.

We create three types of SMSes, namely CONNECT, ACK and DATA, which have different roles in the communication. Figures 9 and 10 show these SMS types in use by our platform. When a CONNECT SMS is sent by client module, the gateway responds with an ACK SMS, allowing the client to send a defined number of DATA SMS. The data portion of this SMS is Base64 encoded. When all the DATA SMS are received by the gateway module, a complete HTTP POST is done. The structure of each type of SMS is described below:

- CONNECT: CONNECT # Process ID # IMEI # SMS Number # URL
- ACK: ACK # Process ID # Gateway Number
- DATA: DATA # Process ID # Sequential Number # Data Portion

The user must inform the gateway number at the client module. When the gateway module receives the first SMS, it parses the SMS and extracts the origin number (SMS Number).

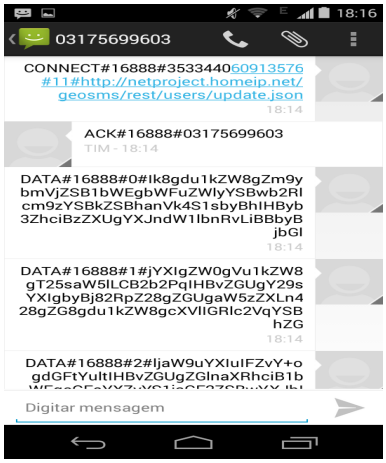

18:16
03175699603
CONNECT#16888#3533440 60913576 #11#http://netproject.homeip.net/ geosms/rest/users/update.json
18:14
ACK#16888#03175699603
TIM - 18:14
DATA#16888#0#Ik8gdu1kZW8gZm9y bmVjZSB1bWEgbWFuZWlyYSBwb2Rl cm9zYSBkZSBhanVk4S1sbyBhIHByb 3ZhciBzZXUgYXJndW1lbnRvLiBBbyB jbGl
18:14
DATA#16888#1#jYXIgZW0gVu1kZW8 gT25saW5lLCB2b2PqIHBvZGUgY29s YXIgbyBj82RpZ28gZGUgaW5zZXLn4 28gZG8gdu1kZW8gcXVlIGRlc2VqYSB hZG
18:14
DATA#16888#2#IjaW9uYXIuIFZvY+o gdGFtYuItIHBvZGUgZGlnaXRhciB1b
Digitar mensagem
18:16
03175699603
DATA#16888#6#RvLCB1bSBjYWJl52 FsaG8gZSB1bWEgYmFycmEgbGF0Z XJhbCBjb3JyZXNwb25kZW50ZXMuIE NsaXF1ZSBlbSBJbnNlcmlyIGUgZXNj b2xoY
18:14
DATA#16888#7#SBvcyBlbGVtZW50b 3MgZGVzZWphZG9zIG5hcyBkaWZlc mVudGVzIGdhbGVyaWFzLg0KVGVtY XMgZSBlc3RpbG9zIHRhbWLpbSBha nVkYW0g
18:14
DATA#16888#8#YSBtYW50ZXIgc2V1 IGRvY3VtZW50byBjb29yZGVuYWRvLi BRdWFuZG8gdm9j6iBjbGljYSBlbSBE ZXNpZ24gZSBlc2NvbGhlIHVtIG5vdm 8
18:14
DATA#16888#9#gdGVtYSwgYXMga W1hZ2VucywgZ3LhZmljb3MgZSBlbG VtZW50b3MgZ3LhZmljb3MgU21hcnR BcnQgc+NvIGFsdGVyYWRvcyBwYXJ
Digitar mensagem
4G 1:08
SMSMaps
SMS List
Type here to filter by contact
+27181
To: Myself +553199535798
On: 2014-11-04 13:31:14
Next to:
From: Santander +27181
To: Myself +553199535798
H 11:02
SMSMaps
Call List
Type here to filter by contact
From: Myself +553199535798
To: Seed Intelligence +553125123910
On: 2014-11-04 13:55:07
2014-11-04 13:57:21
Next to:
Rua Cláudio Manoel, 14-142 - Funcionários, Belo Horizonte - State of Minas Gerais, 30140-100, Brazil

Figure 9. SMS Sequence 1

Figure 10. SMS Sequence 2

6. Sample Tool

We developed a sample mobile application in order to validate our SMS-based platform for transferring small data packages.

This tool, SMS and Calls Maps, allows the user to view their history of messages and voice calls marked geographically by latitude and longitude plus time of send/reception on a map, where the user can directly see the message details and content. The user is also able to filter messages and calls by sending regions, perform local search and view usage statistics. These actions can be performed at mobile or at website. In order to see the data on the site is necessary to perform a synchronization step on mobile device. This synchronization originally runs under data network connection, but we also included the SMS-based platform as a synchronize option.

Figure 11. SMS List

Figure 12. Call LIst

The large number of SMS and Voice Calls that a person may send daily may raise problems related to lack of control and knowledge about old messages that may have been deleted and forgotten by users. Mobile phones provide a list of messages and calls, as seen on Figure 11 and Figure 12, including details, such as date, time and destination

number. Yet, instead of simply a list of SMSes and calls, a richer and more meaningful visualization for the user would provide the calls and SMS the user sent and received geographically distributed on a map.

SMS and Call Maps was developed in Java in order to run on mobile devices using Android operating system. The application intercepts all call or SMS events the user sends or receives. In order to allow this interception, the cellphone user must allow specific permissions to our application. These permissions can vary according to the version of the Android operating system installed on phone.

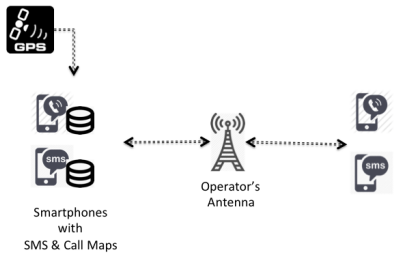


Figure 13. SMS And Call Maps Architecture

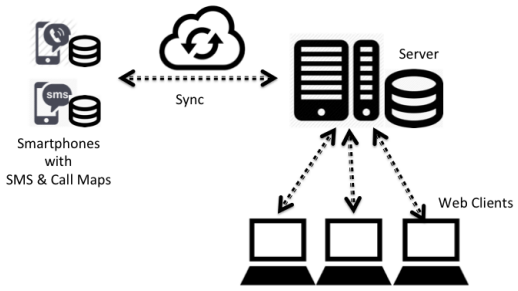


Figure 14. SMS And Call Maps - Sync

Figure 13 shows the architeture of SMS and Call Maps. Geographic coordinates obtained from the GPS running on the cellphone are stored along with the mobile number that originated or received the event as well as the timestamp of the event in a local database. Event timestamp is also stored along with the information mentioned above, in a local database. We need to have access to that information stored in the local database.

We then offer the user the ability to view this information on a website, after it runs a synchronization command in mobile phone. Figures 15 and 16 show screenshots of our application in operation: the first one shows the geographic distribution of all calls and SMS sent/received by the user, whereas the second illustrates a built-in tool that allows the user to view the list of calls and SMS using various filters and export options.

We developed a server module that stores information collected in the mobile devices. This module has been coded in PHP and uses MySQL database. The application allows the user to send the collected data to the server when the mobile device is on the data network coverage. The communication between the android application and PHP server was performed using POST calls, via an API Restfull, as illustrated in Figure 14.

The synchronization between the application and the server module is done over an Internet conection. If this connection is unavailable we use our SMS-based platform to transfer small data packets. Thus, the sync step on figure 14 is replaced by the workflow presented on Figure 8.

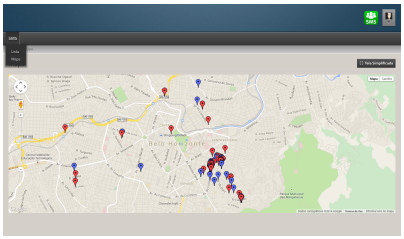


Figure 15. Server Maps

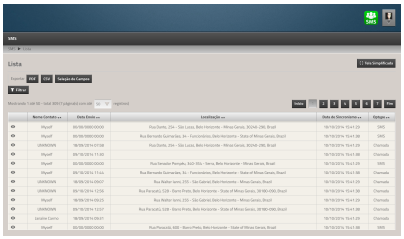


Figure 16. Server Lists

7. Conclusion

This work presents improvements upon the characterization methodology previously proposed by [Xavier et al. 2013]. We included an SMS analysis with the goal of adapting the proposed methodology in order to compare user behavior when using different mobile communication technologies. We analyzed three diferent usage profiles: (i) attendees who made cellphone calls and sent SMS messages, (ii) attendees who only made calls, and (iii) attendees who only sent SMS.

With the analysis of SMS messages and phone calls during soccer matches with high attendance rates, it was possible to identify mobile user behavior patterns when attending such events. We observed that behavior is dependent upon the type of event and usage profile of the mobile device. Voice still prevails as the main means of communication among attendees of these events. The characterization results encouraged us to propose a platform to enrich SMS usage. We present a platform that allows small data packets traffic using SMS. Our main contribution is to keep mobile applications in operation even when the data network is unavailable or congested. We use a cell phone that has Internet access as a gateway. This gateway processes an SMS flow as a request to a web service and returns the results to mobile application also using SMS.

We developed a sample tool in order to validate the proposed platform. SMS and Calls Maps uses a sequence of SMSes to exchange data between mobile application and web server. As future work we expect to share the platform to several developers of mobile applications who rely on the data traffic to keep their applications running when the data network is unavailable or congested. We will also evaluate how to use compression mechanisms along with encryption algorithms.

Acknowledgements

Authors acknowledge CAPES, CNPq, FAPEMIG, FAPERJ, INCTWeb (MCT/CNPq 573871/2008-6) and FIP PUC Minas for their support.

References

Becker, R., Cáceres, R., Hanson, K., Isaacman, S., Loh, J. M., Martonosi, M., Rowland, J., Urbanek, S., Varshavsky, A., and Volinsky, C. (2013). Human mobility characterization from cellular network data. *Commun. ACM*, 56(1):74–82.

Candia, J., González, M. C., Wang, P., Schoenharl, T., Madey, G., and Barabási, A.-L. (2008). Uncovering individual and collective human dynamics from mobile phone records. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224015.

Chen, J., Linn, B., and Subramanian, L. (2009). Sms-based contextual web search. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, MobiHeld '09, pages 19–24, New York, NY, USA. ACM.

Cheung, M. H. and Huang, J. (2013). Optimal delayed wi-fi offloading. In *Modeling & Optimization in Mobile, Ad Hoc & Wireless Networks (WiOpt), 2013 11th International Symposium on*, pages 564–571. IEEE.

Curto, H. N., Ziviani, A., Caetano, J. A., Malab, C. H. S., Almeida, J. M., and Marques-Neto, H. T. (2014). Characterizing the workload dynamics of a mobile phone network during the fifa 2014 world cup. In *Performance Computing and Communications Conference (IPCCC), 2014 IEEE International*, pages 1–2. IEEE.

Eagle, N. and Pentland, A. (2006). Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268.

Eagle, N. and Pentland, A. S. (2009). Eigenbehaviors: Identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066.

Estadao, G. (2013). Estadios da copa terao wi-fi gratis. Website. (2013, Jul 23).

Fjeldsoe, B. S., Marshall, A. L., and Miller, Y. D. (2009). Behavior change interventions delivered by mobile telephone short-message service. *American journal of preventive medicine*, 36(2):165–173.

Gonzalez, M. C., Hidalgo, C. A., and Barabasi, A.-L. (2008). Understanding individual human mobility patterns. *Nature*, 453(7196):779–782.

Lee, K., Hong, S., Kim, S. J., Rhee, I., and Chong, S. (2009). Slaw: A new mobility model for human walks. In *INFOCOM 2009, IEEE*, pages 855–863. IEEE.

Oliver, E. (2010). Characterizing the transport behaviour of the short message service. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 223–238, New York, NY, USA. ACM.

Ratti, C., Williams, S., Frenchman, D., and Pulselli, R. (2006). Mobile landscapes: using location data from cell phones for urban analysis. *Environment and Planning b Planning and Design*, 33(5):727.

Reda, A., Noble, B., and Haile, Y. (2010). Distributing private data in challenged network environments. In *Proceedings of the 19th international conference on World wide web*, pages 801–810. ACM.

Soper, D. (2012). Is human mobility tracking a good idea? *Commun. ACM*, 55(4):35–37.

Veeraraghavan, R., Yasodhar, N., and Toyama, K. (2007). Warana unwired: Replacing pcs with mobile phones in a rural sugarcane cooperative. In *ICTD 2007. International Conference*, pages 1–10. IEEE.

Xavier, F., Malab, C., Silveira, L., Ziviani, A., Almeida, J., and Marques-Neto, H. (2013). Understanding human mobility due to large-scale events. In *Third International Conference on the Analysis of Mobile Phone Datasets (NetMob)*.

Zhuo, X., Gao, W., Cao, G., and Dai, Y. (2011). Win-coupon: An incentive framework for 3g traffic offloading. In *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, pages 206–215. IEEE.

Uma Estratégia de Cache Proativo para Distribuição de Conteúdo em Redes Veiculares

Vitor Borges C. da Silva,
Miguel Elias M. Campista e Luís Henrique M. K. Costa *

[1]GTA/PEE-COPPE/DEL-Poli – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68504 – 21.941-972 – Rio de Janeiro – RJ – Brasil

{borges,miguel,luish}@gta.ufrj.br

Abstract. *This paper explores the use of content oriented networks in the vehicular scenario. In this scenario, the access points installed along the city roads form an integrated wireless access network. This network is used in addition to information regarding the destination of users inside vehicles to schedule and receive in advance the requested content over the access points along their paths to the destination. For this, two strategies to forward interests between access points, consistent with the content oriented paradigm, are developed. These strategies require the implementation of a protocol for neighbor discovery between access points. The results obtained through simulations in urban scenarios show that the joint utilization of network between access points and the proposed strategies increases the fraction of interests satisfied as well as the delivery ratio compared to the standard strategy.*

Resumo. *Este trabalho explora a utilização de redes orientadas a conteúdo no cenário veicular. Nesse cenário, os pontos de acesso instalados pelas vias da cidade formam uma rede integrada de acesso sem fio. Essa rede é utilizada em conjunto com a informação do destino dos usuários dos veículos a fim de agendar e receber antecipadamente os conteúdos solicitados ao longo dos pontos de acesso em suas trajetórias até o destino. Para isso, duas estratégias geográficas de encaminhamento de interesses entre os pontos de acesso, compatível com o paradigma orientado a conteúdo, são desenvolvidas. Essas estratégias requerem a implementação de um protocolo para descoberta de vizinhança entre os pontos de acesso. Os resultados obtidos através de simulações em cenários urbanos mostram que o uso da rede entre os pontos de acesso em conjunto com as estratégias propostas aumenta a fração de interesses atendidos assim como a taxa de entrega em comparação à estratégia padrão.*

1. Introdução

A adoção das redes veiculares em larga escala ainda é um desafio. Como consequência, a infraestrutura de rede é composta por ilhas de conectividade delimitadas pelo alcance do rádio de cada ponto de acesso (AP). Para agravar ainda mais o problema, o movimento dos nós pode tornar as conexões curtas e intermitentes devido a mudanças na topologia e à velocidade dos veículos. Essas características representam uma barreira

*Este trabalho foi parcialmente financiado pela CAPES, CNPq, FAPERJ e FINEP.

ao uso de protocolos tradicionais como o TCP, já que não há garantias da existência de caminhos fim-a-fim [da Silva et al., 2013]. Nesses cenários, as redes tolerantes a atrasos e desconexões (*Delay and disruption Tolerant Networks* – DTN) são frequentemente propostas [da Silva et al., 2013], já que utilizam comutação de mensagens com transferência de custódia. Essa alternativa contorna o problema da conectividade, mas ainda requer que os nós da rede saibam a priori a que nó solicitar um dado conteúdo. Além disso, não considera a possibilidade da coexistência de múltiplas fontes para o mesmo conteúdo ou de múltiplos gateways à Internet, tornando o acesso ineficiente.

As redes orientadas a conteúdo (ROC) [de Brito et al., 2012] são uma alternativa às redes DTN no cenário veicular. Nessas redes, o cliente especifica qual conteúdo deseja, sendo responsabilidade da rede encaminhar o conteúdo ao cliente. O foco está em recuperar um conteúdo, e não em estabelecer uma comunicação com outro nó. Dessa forma, as redes ROC eliminam a necessidade de identificar os nós da rede [Wang et al., 2012a], como feito pelo seu principal representante, a arquitetura *Content Centric Networking* (CCN) [Jacobson et al., 2009]. Uma das características da CCN é a persistência de conteúdos nos nós da rede, similar à vista nas DTNs. Portanto, a CCN pode ser considerada uma arquitetura promissora para o cenário veicular e é considerada neste trabalho.

No escopo de distribuição de conteúdo na Internet, um grande problema do cenário veicular é o curto intervalo de tempo de contato entre os nós da rede, que muitas vezes impossibilita que os conteúdos sejam recebidos pelos veículos. Dessa forma, é importante que as chances de obtenção dos conteúdos solicitados durante a trajetória dos veículos sejam aumentadas, mesmo através de múltiplas fontes. Em outras palavras, um conteúdo solicitado deve ser recebido mesmo que o nó veicular saia da área de cobertura do AP ao qual realizou a primeira solicitação. O uso da trajetória dos veículos em conjunto com o paradigma de redes CCN ainda não foi explorado na literatura.

Este trabalho explora a utilização de cache proativo nos APs para aumentar a probabilidade de entrega de conteúdos no cenário veicular. Em linhas gerais, a ideia consiste em, sabendo o destino geográfico do usuário, encaminhar os interesses dos usuários para os APs em que eles provavelmente irão passar em seu trajeto até o destino. Assim, um AP proativamente solicita os conteúdos requisitados pelos usuários para a Internet antes mesmo que os veículos cheguem em seu alcance. Portanto, ao entrar no alcance de um AP, o conteúdo do usuário já estará disponível. Para atingir o objetivo, neste trabalho são propostas duas estratégias de encaminhamento de interesses veiculares entre os APs e um protocolo de descoberta de vizinhança que provê as informações necessárias para o funcionamento das estratégias. As estratégias de encaminhamento são avaliadas em dois cenários veiculares quanto à fração de interesses dos usuários atendidos, a taxa de entrega e a distância necessária para que um interesse seja atendido. Os resultados mostram que com o aumento da carga da rede as estratégias de encaminhamento aumentam ainda mais seus benefícios, alcançando ganhos de até 50% no número de interesses atendidos.

Este trabalho está organizado da seguinte forma. A Seção 2 apresenta detalhes do uso da CCN no cenário veicular bem como os trabalhos relacionados. A Seção 3 detalha as estratégias de encaminhamento de interesses e o protocolo de descoberta de vizinhança desenvolvidos. Na Seção 4, os cenários de teste são descritos e os resultados obtidos são apresentados. Finalmente, a Seção 5 conclui este trabalho e apresenta as direções futuras.

2. CCN nas Redes Veiculares

Na arquitetura CCN, os nós da rede disseminam pacotes de interesse para solicitar os conteúdos que desejam. Ao receber um interesse, um nó pode encaminhá-lo a outros nós, caso não possua o conteúdo especificado no interesse; ou enviar o conteúdo solicitado no pacote de interesse, caso o possua armazenado. Nota-se que uma característica da CCN é a persistência de conteúdos. Contudo, como o paradigma orientado a estação não é utilizado, as réplicas dos conteúdos armazenados nos nós da rede podem ser utilizadas para atender quaisquer clientes. Os benefícios do armazenamento e replicação de conteúdos de forma descentralizada no cenário veicular são investigados na literatura [Bai e Krishnamachari, 2010]. Além disso, armazenar réplicas de conteúdo na rede é uma forma conhecida de acelerar a obtenção de dados, já que cópias de um conteúdo existem mais próximas dos nós interessados.

O emprego da arquitetura CCN no cenário veicular requer adaptações, já que não foi projetada para o cenário sem fio. Por exemplo, o encaminhamento de interesses pela mesma interface de recepção não faz sentido em redes cabeadas. Contudo, no cenário sem fio, é comum que alguns nós possuam apenas uma interface de rede fazendo com que essa restrição deva ser descartada. Outro problema do uso da arquitetura CCN no meio sem fio é a tempestade de broadcasts (*broadcast storm*), que ocorre quando uma rede é congestionada por retransmissões de pacotes enviados em *broadcast*. No caso da CCN em redes veiculares, isso ocorre devido à política de encaminhamento de pacotes de interesse, que define que um nó que não possua o conteúdo pedido deve reencaminhar o pacote de interesse. Cientes do problema, os trabalhos que utilizam CCN no cenário veicular usam temporizadores para controlar o acesso ao meio de forma que nós próximos não enviem um mesmo pacote de interesse em um curto intervalo de tempo [Prates e Moraes, 2014, Wang et al., 2012a, Amadeo et al., 2013]. Assim, os nós agendam a transmissão de interesses e escutam o canal durante um tempo aleatório. Caso o nó escute o interesse agendado sendo transmitido por outro nó, o agendamento é cancelado. O mesmo vale para os pacotes de conteúdo, evitando tanto o problema de tempestade de broadcasts quanto retransmissões desnecessárias. Por outro lado, o benefício do uso da CCN em um meio de difusão é que uma transmissão pode ser recebida por múltiplos nós, economizando banda passante. Além disso, caso os nós realizem armazenamento promíscuo de conteúdos, ou seja, armazenem conteúdos mesmo que não estejam interessados, a disponibilidade de conteúdos na rede pode aumentar sem custo adicional.

A proposta do uso da arquitetura CCN no cenário veicular é recente e justificada pelos benefícios que o paradigma de orientação a conteúdo traz para o cenário veicular em relação ao antigo paradigma orientado a estação. Os primeiros trabalhos na área [Wang et al., 2012b, Amadeo et al., 2012a] investigaram a viabilidade da substituição do IP pela arquitetura CCN no cenário veicular. Em [Wang et al., 2012b], os autores desenvolveram um esquema de nomeação de conteúdos nas redes veiculares, mostrando uma aplicação de disseminação de informação de tráfego como caso de uso. No trabalho, os autores limitam a um salto o alcance dos interesses enviados, de maneira que os interesses não são encaminhados pelos veículos. Além disso, os autores não consideram a utilização de APs para propagar a informação. Assim, a única forma de um veículo receber uma informação é encontrando um veículo agindo como mula de dados.

Já em [Amadeo et al., 2012a], os autores propõem a arquitetura *Content-Centric*

Vehicular Networking (CCVN), baseada na CCN com modificações para adequá-la ao cenário veicular. Na arquitetura CCVN, os pacotes de interesses são divididos em duas categorias, B-Int (*Basic Interest*) e A-Int (*Advanced Interest*). O B-Int é enviado para receber a primeira parte de um conteúdo formado por múltiplas partes, enquanto os nós descobertos que possuam a informação são identificados e armazenados na estrutura *Content Provider Table* (CPT). Por sua vez, um A-Int é enviado a um dos nós cadastrados na CPT para solicitar as partes restantes do conteúdo. Ao identificar nós da rede de quem as partes restantes do conteúdo devem ser obtidas, essa proposta se aproxima do paradigma orientado a estação. A CCVN remove a FIB (*Forward Information Base*) da arquitetura CCN sob a justificativa de que os nós possuem apenas uma interface de rede, o que nem sempre é verdade. Por exemplo, os APs geralmente possuem mais de uma interface. O trabalho compara o desempenho da arquitetura CCVN com o TCP/IP no cenário veicular através de simulação e avalia as métricas de taxa de entrega e atraso de entrega. Os resultados mostram que a arquitetura baseada no CCN supera o TCP/IP no cenário veicular.

Em [Wang et al., 2012a], é desenvolvida uma aplicação de disseminação de informação de tráfego que usa o arcabouço de nomeação de conteúdos de [Wang et al., 2012b]. Como o meio sem fio não permite detecção de colisões, os autores propõem temporizadores para minimizar a chance de colisões. O conjunto de temporizadores é formado por quatro distintos: *collision-avoidance timer*, *pushing timer*, *NDN-layer retransmission timer* e *Application retransmission timer*. Desses quatro, apenas o primeiro e o último são usados nesse trabalho. O primeiro cancela a transmissão de pacotes caso um nó ouça outro transmitindo o mesmo pacote. O segundo prioriza a transmissão de nós mais distantes do produtor da informação, o que é específico à aplicação proposta em [Wang et al., 2012a]. O terceiro retransmite uma mensagem em camadas inferiores, enquanto o quarto retransmite interesses não atendidos na camada de aplicação.

O trabalho [Prates e Moraes, 2014] propõe um arcabouço para reduzir o problema do *broadcast storm* de pacotes de interesse da CCN no cenário veicular. O arcabouço é composto por um esquema de nomeação de conteúdos geo-referenciado e um mecanismo de encaminhamento de pacotes de interesse. O arcabouço é implementado no ndnSIM e comparado com o trabalho [Wang et al., 2012a], e com o CCN puro. Os resultados mostram que definir escopos geográficos aos pacotes de interesse em conjunto com o uso dos temporizadores reduzem ainda mais a tempestade de broadcast [Prates e Moraes, 2014].

Apesar de existirem trabalhos que propõem a utilização da CCN em redes veiculares, nesses trabalhos os APs não são explorados de forma a aumentar a taxa de entrega de conteúdos [Prates e Moraes, 2014, Wang et al., 2012a, Amadeo et al., 2013, Wang et al., 2012b, Amadeo et al., 2012b]. Além disso, nenhuma informação da trajetória do veículo ou do seu destino é usada para tornar a rede ciente do destino. Este trabalho preenche essas lacunas, como visto a partir da próxima seção (Seção 3).

3. Estratégias para Distribuição de Conteúdo em Redes Veiculares

Este trabalho propõe o uso de cache proativo nos APs para melhorar a distribuição de conteúdos no cenário veicular. A ideia básica consiste em distribuir *proativamente* o conteúdo solicitado por cada usuário ao longo dos APs em sua trajetória até o destino final, de forma que o conteúdo esteja disponível antes do usuário chegar. Essa estratégia aumenta a eficiência da rede já que solicitar e receber o conteúdo no mesmo AP não é

trivial dado o tempo de contato entre veículos e APs. A ideia pode ser vista na Figura 1.

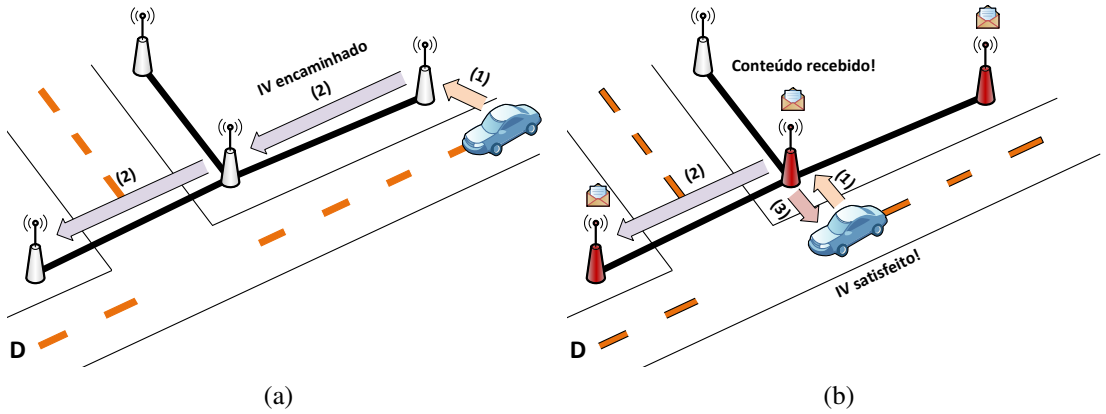


Figura 1. Ideia da proposta: (a) Solicitação: a seta em (1) representa o pedido de um conteúdo, já as setas em (2) representam o encaminhamento do pedido aos APs na direção do destino D; (b) Disponibilidade: os APs com conteúdo acima já podem entregar ao usuário as suas requisições. A seta em (3) representa a entrega do conteúdo.

Para conhecer a trajetória do usuário, inserem-se campos no cabeçalho dos pacotes de interesse veiculares (IVs) para armazenar a posição geográfica atual do usuário, o seu destino e o horário estimado de chegada ao destino. Além disso, assume-se que os APs conhecem a própria posição geográfica e a dos APs vizinhos de um salto. A descoberta de vizinhança é feita por um protocolo de descoberta de vizinhança proposto. Ao conhecer sua posição e a de seus vizinhos, os APs podem encaminhar a outros APs os IVs encapsulados em pacotes de interesse de ponto de acesso (IAPs). Assim, o conteúdo de um usuário pode ser solicitado à Internet nos APs onde é previsto que ele passe, antes que ele chegue lá. A posição atual, o destino e o tempo de chegada até o destino do usuário podem ser obtidos dos GPSs presentes nos veículos ou nos smartphones dos usuários. Caso um usuário não deseje informar o seu destino ele ainda será capaz de usar a rede, mas sem os benefícios do encaminhamento.

Nas CCNs o roteamento e identificação de pacotes se baseiam nos nomes. Assim, para propor estratégias de roteamento e identificar os pacotes na rede devem ser definidos esquemas de nomeação para interesses e conteúdos que realizam o roteamento desejado e sejam hierárquicos. Dessa forma, neste trabalho um IV segue o esquema de nomeação: $\backslash iv \backslash idConteudo \backslash idParte$. O primeiro componente define o escopo veicular. O segundo componente identifica o conteúdo solicitado pelo usuário, enquanto o terceiro identifica as partes que constituem esse conteúdo. Ao solicitar-se um conteúdo, são enviados IVs para cada uma das partes que formam o conteúdo pedido. Da mesma maneira, define-se o esquema de nomeação dos IAPs: $\backslash iap \backslash idProtocolo \backslash idInterfaceDeSaída \backslash idAP \backslash idIAP$. O primeiro componente do nome define o escopo do IAP, restringindo seu uso à rede entre os APs. O segundo componente indica a qual protocolo ou estratégia um IAP pertence, por exemplo, uma estratégia de encaminhamento de interesse ou ao protocolo de descoberta de vizinhança. O terceiro componente define a interface de saída pela qual um IAP é encaminhado, o quarto indica o AP que criou o IAP e o quinto identifica o próprio IAP.

Para descrever como os interesses são encaminhados nas CCNs é necessário des-

crever como as FIBs dos nós são configuradas. As FIBs dos veículos possuem uma entrada (*iv*) responsável pelo encaminhamento dos IVs pela interface sem fio. Já as FIBs dos APs possuem além da entrada para encaminhar e receber IVs pela interface sem fio, duas entradas para cada interface de rede que conecta um AP a outro AP vizinho, ambas no formato *iap**idProtocolo**idInterfaceDeSaída**idAP*. Isso muda o terceiro componente a cada interface de saída sendo relacionada na FIB. Uma das entradas é para o encaminhamento dos IAPs do protocolo de descoberta da vizinhança, a outra é específica à estratégia de encaminhamento de IVs. O uso do identificador dos APs nos IAPs junto com as FIBs dos APs faz com que cada IAP seja encaminhado apenas por um salto, o que está de acordo com as estratégias de roteamento salto a salto desenvolvidas. Definidos o esquema de nomeação utilizado e a população das FIBs dos nós na rede, define-se a seguir como os usuários nos veículos solicitam conteúdos.

3.1. Solicitação e Recebimento de Conteúdos pelos Usuários

Qualquer conteúdo maior que uma parte de conteúdo é fragmentado em múltiplas partes que juntas formam o conteúdo original. Quando um usuário deseja um conteúdo, por exemplo, o jornal diário, a aplicação envia IVs pelas partes do conteúdo às camadas inferiores, onde uma entrada na PIT (*Pending Interest Table*) para cada IV é criada, e onde agenda-se a transmissão dos IVs pela rede sem fio de acordo com a FIB. Como no cenário veicular podem não existir nós próximos que escutem uma solicitação, ou nós que possuam conexão com a Internet ou ainda nós que possuam o conteúdo na CS (*Content Store*), torna-se necessária a presença de uma lista para retransmissão dos IVs não atendidos após um determinado intervalo de tempo. Logo, cada IV solicitado é inserido nessa lista de retransmissão com o horário do último envio, para controlar o momento da próxima retransmissão. Quando um IV é retransmitido a hora estimada de chegada ao destino é atualizada, reduzindo a influência do comportamento do usuário ou do trânsito.

Ao um pacote ser recebido pela interface sem fio do usuário, verifica-se se o pacote é um interesse ou um conteúdo. Após, verifica-se se existe um agendamento para transmitir esse mesmo pacote na interface sem fio. Em caso afirmativo a transmissão agendada é cancelada. Se o pacote recebido for um IV, verifica-se se ele já está registrado na PIT; se não estiver, cria-se uma entrada na PIT para ele e busca-se na CS a parte do conteúdo que o satisfaz. Caso a parte do conteúdo seja encontrada, a transmissão dela pela interface sem fio é agendada; caso contrário, o IV é encaminhado de acordo com a FIB. Já no caso de o IV recebido já estar registrado na PIT, apenas atualiza-se a entrada do IV na PIT, zerando o contador que expira a entrada correspondente.

Se o pacote recebido for uma parte de um conteúdo, verifica-se se ele foi solicitado existindo uma entrada na PIT para aquela parte do conteúdo. Em caso positivo, se a parte do conteúdo recebida não existir na CS, ela é adicionada à CS e a sua retransmissão é agendada na interface sem fio. Em seguida, a entrada na PIT referente ao conteúdo é removida e o conteúdo é encaminhado à aplicação que o solicitou. Na aplicação, o IV referente à parte do conteúdo recebida é removido da fila de retransmissão e o conteúdo é remontado e entregue ao usuário caso todas as partes tenham sido recebidas. Contudo, se a parte do conteúdo não foi solicitada, apenas armazena-se a parte do conteúdo na CS. Isso caracteriza o armazenamento promíscuo de conteúdos por armazenar conteúdos não solicitados. Vale notar que somente conteúdos solicitados são retransmitidos para evitar possíveis congestionamentos no meio sem fio causado por conteúdos não solicitados.

Definido como os usuários interagem com a rede, é possível descrever como os APs atuam para entregar os conteúdos da Internet aos usuários.

3.2. Recebimento e Encaminhamento de IVs

Sem o uso do cache proativo proposto, os APs somente solicitam conteúdos da Internet pedidos diretamente pelos usuários através da interface sem fio. Essa estratégia, aliada ao tempo curto de contato entre veículos e APs, prejudica a entrega de conteúdos ao usuário. A proposta é então fazer com que os APs encaminhem os pedidos dos usuários aos outros APs. Encaminhar cada IV a todos os APs seria ineficiente. Sendo assim, os APs devem encaminhar os IVs apenas para os APs em regiões onde o usuário que solicitou o conteúdo for provavelmente passar. É possível que os IVs sejam encaminhados para regiões onde o veículo não passará, já que se assume conhecimento apenas da posição atual e do destino do usuário e não da trajetória completa até o destino. Contudo, a retransmissão de IVs não atendidos fornece uma nova chance para que o caminho feito pelo veículo seja previsto pelas estratégias de encaminhamento. Isso ocorre, pois a posição do usuário muda com o tempo e, portanto as duas abordagens podem prever caminhos diferentes para um IV de um usuário com o passar do tempo.

Ao receber um IV pela interface sem fio um AP agenda a retransmissão do IV, extrai as informações inseridas no cabeçalho e adiciona o IV em uma estrutura de dados que armazena as solicitações pendentes de conteúdos da Internet. Em seguida, o AP encaminha as informações do IV à estratégia de encaminhamento que vai enviar o IV encapsulado em um IAP a outros APs. Vale notar que os IAPs de encaminhamento, apesar de serem pacotes de interesse, não são respondidos por não existirem conteúdos associados a eles. Apenas os IVs e os IAPs do protocolo de descoberta de vizinhança possuem conteúdos associados e, portanto são respondidos. Para encaminhar IVs entre APs, os APs precisam conhecer seus vizinhos. Para tal, um protocolo de descoberta de vizinhança que usa o paradigma orientado a conteúdo foi desenvolvido.

Protocolo de Descoberta de Vizinhança: O protocolo de descoberta de vizinhança permite que cada AP conheça a posição de seus vizinhos e a interface pela qual esse vizinho pode ser encontrado. Essas informações, em conjunto com as configurações nas FIBs, permitem que o encaminhamento de IVs feito pelos APs seja direcionado geograficamente. O protocolo executa apenas uma vez, quando o AP é ligado, enviando IAPs de descoberta de vizinhança em todas as suas interfaces com outros APs. Ao receber um pacote de interesse, cada AP verifica através do nome se ele é do protocolo de descoberta de vizinhança. Se ele for um interesse de descoberta, o AP responde o IAP de descoberta com um conteúdo com o mesmo nome do interesse original, contendo o próprio identificador e a sua posição geográfica. Já quando recebe um conteúdo como resposta ao IAP de descoberta, o AP acrescenta o novo vizinho à lista de vizinhos. Para cada novo vizinho, o AP registra a identificação e posição geográfica a partir da carga do conteúdo, e a interface de saída usada para alcança-lo a partir do nome do conteúdo. Esse nome é igual ao do interesse enviado pelo próprio AP para descobrir o vizinho em uma interface. Uma vez descoberta a vizinhança, os APs são capazes de decidir a qual AP vizinho os IAPs com IVs encapsulados devem ser encaminhados.

Encaminhamento de IVs em Área Geográfica Triangular: A ideia do triângulo de cobertura é delimitar a área onde provavelmente o veículo irá passar em seu trajeto até o

destino. A escolha de um triângulo como figura delimitadora se justifica pelo triângulo aumentar a área em seu interior com o aumento da distância da origem. O conceito se encaixa no cenário veicular já que a probabilidade de acertar a posição futura de um veículo é maior quanto mais perto estiver do último ponto onde foi visto. Como consequência, o número de APs em que se espera que um usuário passe até o destino aumenta com a distância desde o último ponto onde ele foi visto.

Ao receber um IV pela interface sem fio, as informações de posição atual, destino e hora de chegada ao destino do usuário que enviou o IV são usadas pelo AP para criar o triângulo de cobertura desse IV. Um triângulo de cobertura é definido pela posição do AP que deu origem ao triângulo R, que é um dos vértices do triângulo; a posição de destino do veículo que enviou o interesse D; e um ângulo de abertura α, que define os ângulos $\angle D1RD$ e $\angle DRD2$. Vale notar que o triângulo de cobertura é isósceles, com os lados $D1R$ e $D2R$ iguais. A partir dessas informações, calcula-se a posição dos outros dois vértices, $D1$ e $D2$, que formam o triângulo $\triangle RD1D2$.

Definido o triângulo de cobertura, o AP insere os vértices que formam o triângulo e o IV com todas as suas informações em um IAP de encaminhamento e inicia o processo de encaminhamento dele aos outros APs. O processo de encaminhamento de IVs na área triangular se baseia em duas regras: se um AP está dentro do triângulo de cobertura, ele encaminha o IAP a toda a sua vizinhança. Caso contrário, ele encaminha o IAP apenas aos seus vizinhos de 1 salto que estejam dentro do triângulo de cobertura. Isso é possível, pois um AP conhece o triângulo e a posição geográfica de seus vizinhos.

O processo de encaminhamento é iniciado pelo AP que recebe o IV pela interface sem fio, e que é o ponto R do triângulo de cobertura. Esse AP por definição faz parte do triângulo de cobertura e encaminha o IAP a todos os seus vizinhos. Um AP recebendo o IAP com o IV encaminhado e estando dentro do triângulo de cobertura contido no IAP além de continuar o processo de encaminhamento, adiciona o IV encapsulado na estrutura de dados que guarda as solicitações de conteúdos da Internet pendentes. Isso é feito, pois o AP se encontra dentro da área onde é esperado que o usuário passe no trajeto até o destino. Na estratégia de encaminhamento descrita, os APs podem receber réplicas dos IAPs encaminhados, para impedir que o encaminhamento continue indefinidamente cada AP possui uma lista com os últimos K IAPs recebidos. Assim, a cada IAP recebido o AP verifica se ele está na lista, e somente em caso negativo, o IAP recebido é encaminhado.

A estratégia de encaminhamento dentro do triângulo de cobertura tem a vantagem de ser abrangente, contudo introduz réplicas de IAPs. A seguir define-se a estratégia de encaminhamento de IVs por Minimização de Distâncias Salto-a-Salto que apesar de menos abrangente, introduz menos IAPs na rede entre APs e não introduz réplicas.

Encaminhamento de IVs por Minimização de Distâncias Salto-a-Salto: O objetivo dessa estratégia é restringir o número de APs aos quais os IAPs são encaminhados, e consequentemente diminuir o número de APs onde um conteúdo solicitado é inserido na estrutura de pedidos pendentes da Internet. A redução de encaminhamentos ocorre devido à ausência de réplicas e devido a essa estratégia encaminhar IAPs para no máximo uma interface. Ao receber um IV pela interface sem fio, a partir da posição do destino do usuário, o AP escolhe a qual AP ele vai encaminhar o IAP com o IV encapsulado. Para isso, o AP seleciona o seu vizinho cuja distância até o destino é a menor. Se o vizinho

selecionado estiver mais perto do destino que o próprio AP, o IAP é encaminhado ao vizinho selecionado. Caso contrário, o processo de encaminhamento termina. Um AP ao receber um IAP encaminhado realiza o mesmo procedimento para continuar ou parar o encaminhamento. Diferentemente da estratégia de encaminhamento do triângulo de cobertura, todos os nós que recebem o IAP inserem o IV encapsulado na estrutura de dados que armazena as solicitações pendentes de conteúdos da Internet.

3.3. Recebimento e Encaminhamento de IVs e Recebimento de Conteúdos da Internet pelos APs

Um AP pode receber uma parte de um conteúdo pela interface sem fio, caso ele escute a transmissão, ou ainda, caso um usuário com a parte do conteúdo esteja ao alcance do AP no momento que o IV é retransmitido pelo AP. Além disso, um AP pode solicitar e receber partes de conteúdos da Internet. Para ambos os casos o AP ao receber o uma parte de conteúdo realiza o fluxograma apresentado na Figura 2.

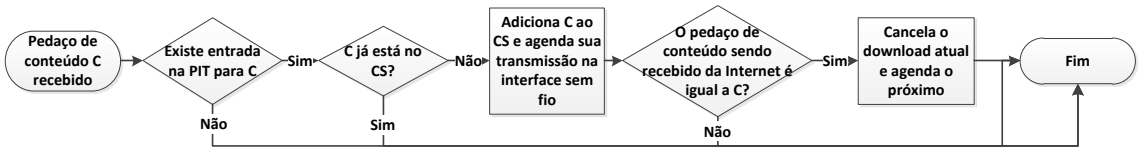


Figura 2. Fluxograma do processo de recebimento de uma parte de conteudo pelo AP.

Vale ressaltar que para uma entrada na PIT para uma parte de conteúdo existir, um IV deve ter sido enviado ao AP diretamente por um veículo. Os IVs encaminhados dentro dos IAPs, apesar de serem solicitados à Internet, não criam entradas na PIT. Nesses casos quem cria a entrada na PIT é o IAP de encaminhamento que encapsula o IV. Isso é feito para que as partes de conteúdo que sejam recebidas da Internet em pontos onde não é garantido que o veículo esteja não sejam transmitidas no meio sem fio. Dessa forma, as estratégias de cache proativo utilizadas recebem os conteúdos nos APs antes que o usuário chegue lá, mas para que o usuário receba o conteúdo, ainda é necessário que ele envie um IV diretamente a um AP ou veículo que possua o conteúdo solicitado. Na Seção 4 as propostas são avaliadas em dois cenários veiculares.

4. Resultados

As estratégias de encaminhamento propostas foram implementadas no simulador ndnSIM e avaliadas em dois cenários veiculares, criados no software SUMO (*Simulation of Urban MObility*). As propostas foram comparadas com o CCN modificado para atuar no cenário veicular, sendo o encaminhamento de IVs a única diferença entre a proposta e o CCN usado. As vias possuem velocidade máxima de 50 km/h em ambos os cenários. O cenário rodoviário é composto de duas pistas duplas de 6 km de comprimento em sentidos opostos. Entre as pistas há um espaço de 100 metros, no qual os APs são uniformemente distribuídos ao longo da via. O cenário urbano é formado por uma matriz 15 por 15 de quarteirões de 150m x 150m, totalizando 2.250 metros x 2.250 m. Os APs são distribuídos uniformemente formando uma malha igualmente espaçada nos eixos X e Y.

As interfaces de rede sem fio utilizadas na simulação são do padrão 802.11a operando no modo ad hoc na taxa de 24 Mb/s com modulação OFDM, em substituição ao

padrão 802.11p não presente no simulador usado. O canal sem fio é configurado com modelo de atraso de propagação de velocidade constante, modelo de propagação três-log-distância e o modelo de propagação Nakagami-m. As interfaces ponto-a-ponto entre os APs possuem taxa de 5 Mb/s e atraso constante de 2 ms. Cada AP está conectado à Internet com um enlace de 256 kb/s . Os IVs possuem TTL igual a 1 para evitar o *broadcast storm* em conjunto com o temporizador de [Wang et al., 2012a] que evita colisões. Além disso, os IVs expiram após 10 segundos na PIT e são retransmitidos pela aplicação após 11 segundos, caso não tenham sido satisfeitos. Na simulação existem 1.000 conteúdos que são divididos em partes de 1.500 bytes. Os usuários solicitam um novo conteúdo em média a cada 10 segundos de acordo com uma variável aleatória exponencial [Chlebus e Brazier, 2007]. Os conteúdos solicitados seguem uma distribuição Zipf com parâmetro 0,8, essa distribuição representa a popularidade dos conteúdos de acordo com as análises de traços empíricos em [Breslau et al., 1999]. Todos os nós da rede realizam cache promíscuo armazenando conteúdos mesmo sem interesses registrados. Contudo, somente os APs com a proposta realizam cache proativo, solicitando conteúdos à Internet ao longo da rota dos usuários.

Nos gráficos, a sigla EMD se refere ao encaminhamento por minimização de distâncias, a sigla ET ao encaminhamento em área triangular e CCN é o CCN modificado para o cenário veicular sem encaminhamento. Nos gráficos onde o número de APs varia, a legenda é apresentada no formato "sigla - número de APs". Já nos gráficos onde o tamanho dos conteúdos é variado, a legenda segue o formato "sigla - tamanho do conteúdo" em kB. É utilizada a média de 30 experimentos com intervalo de confiança de 95%.

4.1. Cenário Rodoviário

Os APs possuem uma estrutura de dados que regula a requisição de conteúdos da Internet. No cenário rodoviário, a proposta utiliza uma fila com prioridade, enquanto o caso convencional utiliza uma fila comum. Quando o encaminhamento de IVs é utilizado, mais solicitações de conteúdo vão chegar a um AP, já que além dos pedidos dos usuários em seu alcance ele ainda recebe os encaminhados. A fila com prioridade serve para orquestrar as solicitações de conteúdos à Internet, sendo a prioridade definida como o menor horário estimado de chegada até o AP (HCAP). Assim, os pedidos de um usuário que deve chegar mais cedo ao AP terão maior prioridade. O HCAP é estimado a partir das informações contidas no IV da seguinte forma:

$$HCAP = \left[\left(\frac{dist\,(pU, pAP)}{dist\,(pU, pDest)} \right) * (hDest - hAtual) \right] + hAtual, \qquad (1)$$

onde pU é a posição atual do usuário, $pDest$ seu destino, $hDest$ o horário estimado de chegada do usuário ao destino, $hAtual$ o horário atual no AP, pAP a posição do AP e $dist()$ uma função que calcula a distância entre dois pontos. Se em algum momento um pedido possuir HCAP inferior à hora atual ele é removido da fila de pedidos do AP, pois provavelmente o usuário interessado já passou. Vale notar que como o cenário define uma rota obrigatória em linha reta para os veículos, não existe a possibilidade de um usuário não passar por um AP onde seu conteúdo foi agendado. Dessa forma, as competições entre IVs solicitados por um usuário no alcance do AP e IVs encaminhados afetam menos o número de IVs satisfeitos.

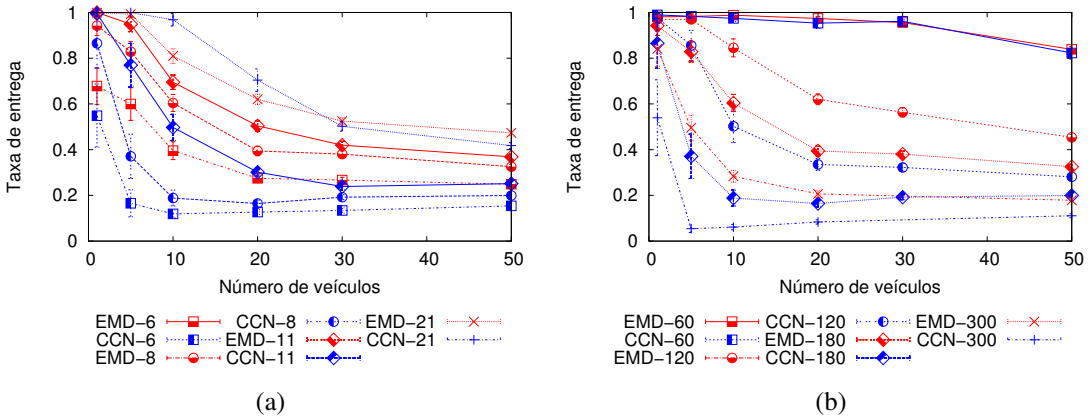


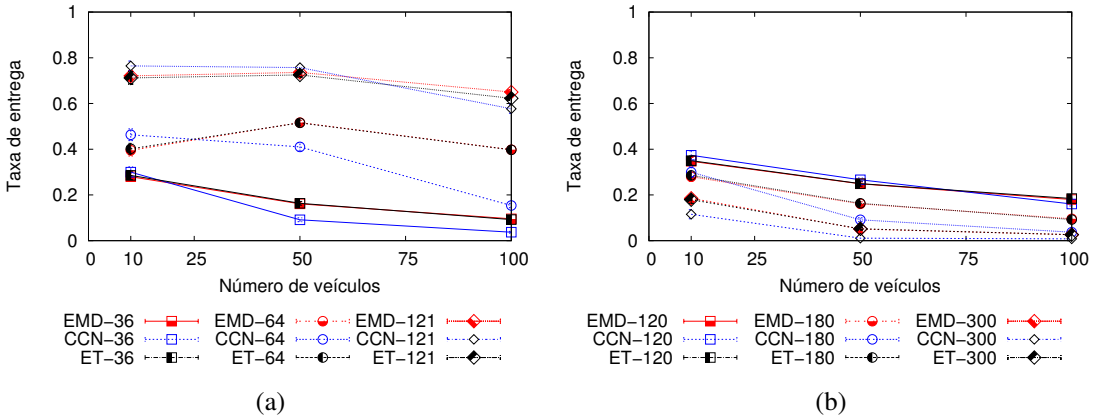
Figura 3. Taxa de entrega de conteúdos completos: (a) Diferente número de APs com conteúdos de tamanho 180 kBytes; (b) Diferente tamanho dos conteúdos com 8 APs na via.

Como as estratégias de encaminhamento propostas obtêm o mesmo resultado devido à topologia do cenário, somente um resultado é mostrado. Na Figura 3 compara-se a taxa de entrega de conteúdos completos da proposta com o CCN sem encaminhamento para diferentes números de veículos na via: 1, 5, 10, 20, 30, 50. Na Figura 3(a) o número de APs distribuídos na via varia mantendo o mesmo número de veículos, já que quanto maior o número de APs na via, maior a taxa de entrega em ambos os casos. Na Figura 3(b) o tamanho dos conteúdos solicitados pelos usuários varia e nota-se que mantido o mesmo número de veículos, quanto maiores os conteúdos menores são as taxas de entrega em ambos os casos. Observando as Figuras 3(a) e 3(b) também nota-se que aumentando somente o número de veículos a taxa de entrega diminui. Todos esses resultados podem ser explicados através da relação entre capacidade e carga da rede. Ao aumentarmos a capacidade da rede de receber conteúdos da Internet, por exemplo, aumentando o número de APs como na Figura 3(a), e mantendo a carga constante as taxas de entrega tendem a aumentar. Similarmente, se a carga da rede aumenta, com o número de veículos ou tamanho dos conteúdos, e a capacidade se mantém, as taxas de entrega tendem a diminuir. Apesar disso, percebe-se pelas Figuras 3(a) e 3(b) que o uso da estratégia de encaminhamento faz a taxa de entrega ser menos suscetível ao aumento na carga da rede, portanto quanto maior a carga na rede, mais benéfico o uso do encaminhamento. Um exemplo disso está na Figura 3(a). Na curva de 21 APs, o cenário se aproxima de um cenário totalmente coberto, e a taxa de entrega do CCN modificado começa maior, contudo com o aumento do número de veículos o uso do encaminhamento torna-se a melhor opção.

4.2. Cenário Urbano

Diferente do cenário rodoviário, no cenário urbano existem múltiplas rotas entre dois pontos, de forma que é possível que um usuário não passe por um AP onde seu conteúdo foi agendado. Assim, competições entre IVs solicitados por um usuário no alcance do AP e IVs encaminhados não são desejáveis. Para evitar que esta competição ocorra, a estrutura de dados que regula o acesso à Internet é dividida em duas com prioridades distintas: a estrutura de maior prioridade armazena os IVs recebidos pela interface sem fio, e a de menor prioridade os IVs encaminhados. A estrutura de maior prioridade, dos IVs recebidos diretamente, é similar a uma pilha, contudo caso a estrutura fique cheia

o descarte é realizado no fundo da estrutura. A ideia em utilizar uma pilha é atender primeiro os pedidos mais recentes, pois é mais provável que quem os pediu ainda esteja no alcance. Já a estrutura de menor prioridade é uma fila de prioridade idêntica a do cenário anterior. Para escolher o próximo item a ser solicitado à Internet, o AP escolhe primeiro os itens da estrutura de maior prioridade. Somente no caso dessa estrutura estar vazia é que os itens da estrutura de menor prioridade são agendados para recebimento. No entanto, caso um pedido seja adicionado à fila de maior prioridade durante o recebimento de um pedido da estrutura de menor prioridade, o recebimento em curso não é interrompido.



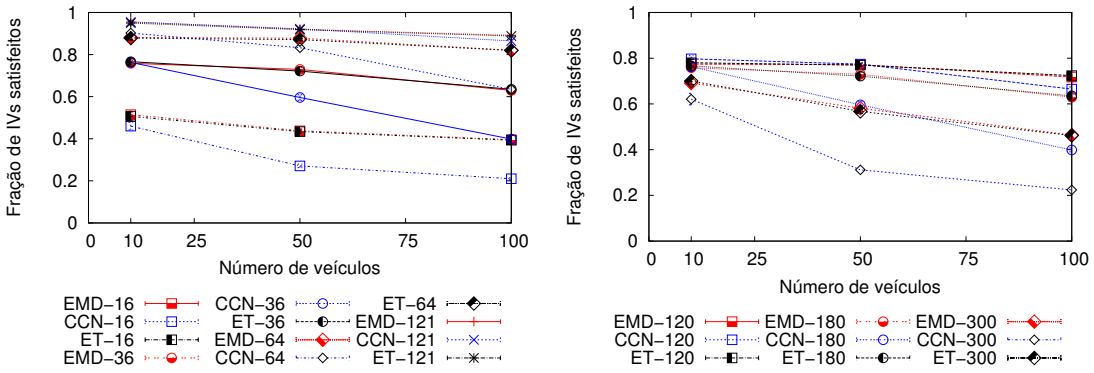
(a) (b)

Figura 4. Taxa de entrega de conteúdos completos: (a) Diferente número de APs com conteúdos de tamanho 180 kBytes; (b) Diferente tamanho dos conteúdos com 36 APs na via.

Nas Figuras 4(a) e 4(b) observa-se o mesmo comportamento do cenário rodoviário: quanto maior a relação carga capacidade na rede mais benéfico é o encaminhamento. Analisando os valores da taxa de entrega isoladamente, poderia-se concluir que o número de APs a ser usado deveria ser igual ou superior a 121 para oferecer o serviço aos usuários. Contudo, vale notar que os APs na cidade devem ser apenas uma das opções para prover o acesso à Internet aos usuários e não a única. Assim, vale a pena avaliar a influência da proposta na fração de IVs satisfeitos, já que outras formas de comunicação poderiam auxiliar na recepção dos IVs dos usuários não satisfeitos pelos APs.

Na Figura 5(a) verifica-se que apesar da taxa de entrega com 36 APs, conteúdos de 180 kB e 100 veículos estar bem abaixo de 20%, a fração de IVs satisfeitos utilizando qualquer encaminhamento ultrapassa os 60%, enquanto a fração de IVs satisfeitos sem o encaminhamento permanece em torno de 40%. Isto representa um ganho de 50% com o uso do encaminhamento proposto, o que permite que menos APs sejam utilizados para prover uma mesma qualidade de serviço aos usuários, reduzindo os custos e consequentemente as barreiras para que a conectividade no cenário veicular seja alcançada. Vale notar que os mesmos comportamentos para a taxa de entrega são observados variando o número de APs ou o tamanho dos conteúdos, como pode ser visto nas Figuras 5(a) e 5(b).

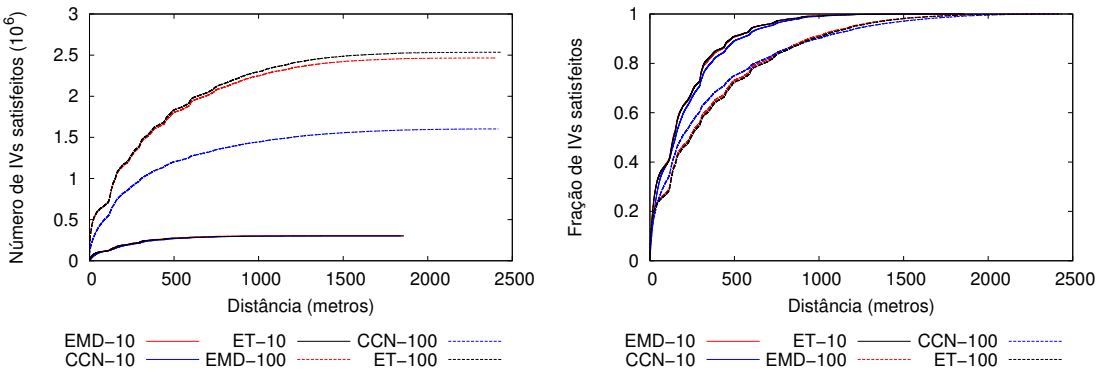
Para avaliar a velocidade com que IVs são satisfeitos, na Figura 6 o número de APs é fixado em 36 e o tamanho dos conteúdos em 180 kB e avalia-se a distância entre os pontos do primeiro envio de um IV e o ponto onde ele é satisfeito. A Figura 6 reúne os IVs satisfeitos nas 30 rodadas de cada proposta para efetuar a comparação. Na Figura 6(a), nota-se que o número de IVs satisfeitos é superior com o uso de qualquer en-



(a) (b)

Figura 5. Fração de IVs atendidos: (a) Diferente número de APs com conteúdos de tamanho 180 kBytes; (b) Diferente tamanho dos conteúdos com 36 APs na via.

caminhamento. Nota-se ainda que o encaminhamento em área triangular por incluir mais APs que o encaminhamento por minimização de distâncias satisfaz um maior número de IVs. Já pelo gráfico normalizado na Figura 6(b), pode-se avaliar a velocidade com que IVs são satisfeitos. Com 10 veículos, observa-se que com o uso dos encaminhamentos uma fração maior de IVs é satisfeita para uma mesma distância. Já para o caso de 100 veículos, a CCN satisfaz uma fração maior de IVs para distâncias até aproximadamente 800 metros. Esse comportamento ocorre, pois o uso do encaminhamento permite que mais IVs sejam satisfeitos a longas distâncias devido ao agendamento pela trajetória, ao passo que o CCN puro não é capaz de satisfazer esses IVs, satisfazendo menos IVs em absoluto (Figura 6).



(a) (b)

Figura 6. IVs satisfeitos de acordo com a distância: (a) Não normalizado; (b) Normalizado pelo número de IVs satisfeitos.

5. Conclusões e Trabalhos Futuros

Este trabalho propôs o uso de cache proativo nos APs para aumentar a probabilidade de entrega de conteúdos no cenário veicular. Para isso, foram propostas duas estratégias geográficas de encaminhamento de interesses veiculares entre os APs e um

protocolo de descoberta de vizinhança. As estratégias de encaminhamento foram avaliadas em dois cenários veiculares quanto à fração de interesses atendidos, a taxa de entrega e a distância necessária para que um interesse seja atendido. Os resultados mostram que, com o aumento da carga da rede, as estratégias de encaminhamento aumentam ainda mais seus benefícios. Com ganhos de até 50% no número de interesses veiculares atendidos, o uso das estratégias de encaminhamento permite que menos APs sejam utilizados para prover a mesma qualidade de serviço aos usuários, facilitando o provimento de conectividade no cenário veicular.

Como trabalhos futuros planeja-se estender os testes para cidades reais, para ver se haveria uma maior diferença nos resultados das estratégias propostas. Além disso, pretende-se avaliar como alguns parâmetros, como o enlace dos APs com a Internet e o ângulo α do triângulo de cobertura, influenciam os resultados.

Referências

Amadeo, M., Campolo, C. e Molinaro, A. (2012a). Content-centric networking: Is that a solution for upcoming vehicular networks? Em *ACM VANET'12*, p. 99–102.

Amadeo, M., Campolo, C. e Molinaro, A. (2012b). Crown: Content-centric networking in vehicular ad hoc networks. *IEEE Communications Letters*, 16(9):1380–1383.

Amadeo, M., Campolo, C. e Molinaro, A. (2013). Enhancing content-centric networking for vehicular environments. *Computer Networks*, 57(16):3222 – 3234.

Bai, F. e Krishnamachari, B. (2010). Exploiting the wisdom of the crowd: localized, distributed information-centric VANETs. *IEEE Communications Magazine*, 48(5):138–146.

Breslau, L., Cao, P., Fan, L., Phillips, G. e Shenker, S. (1999). Web caching and Zipf-like distributions: Evidence and implications. Em *IEEE INFOCOM*, volume 1, p. 126–134.

Chlebus, E. e Brazier, J. (2007). Nonstationary poisson modeling of web browsing session arrivals. *Inf. Process. Lett.*, 102(5):187–190.

da Silva, V. B., da Silva, F. O., Campista, M. E. M. e Costa, L. H. M. K. (2013). Roteamento baseado na trajetória para redes veiculares desconectadas com múltiplos gateways. *SBRC*, p. 1038–1051.

de Brito, G. M., Velloso, P. B. e Moraes, I. M. (2012). Redes orientadas a conteúdo: Um novo paradigma para a internet. *Minicursos do SBRC*, p. 211–264.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H. e Braynard, R. L. (2009). Networking named content. Em *ACM CoNEXT*, p. 1–12.

Prates, A. A. e Moraes, I. M. (2014). Geozone: Um framework eficiente de difusão de interesses em redes veiculares orientadas a conteúdo. *SBRC*, p. 163–176.

Wang, L., Afanasyev, A., Kuntz, R., Vuyyuru, R., Wakikawa, R. e Zhang, L. (2012a). Rapid traffic information dissemination using named data. Em *ACM NoM*, p. 7–12.

Wang, L., Wakikawa, R., Kuntz, R., Vuyyuru, R. e Zhang, L. (2012b). Data naming in vehicle-to-vehicle communications. Em *IEEE NOMEN*, p. 328–333.

Otimização do Posicionamento de Servidores Físicos em Centros de Dados Resilientes a Desastres

Rodrigo S. Couto[1], **Stefano Secci**[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]Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France

{souza,miguel,luish}@gta.ufrj.br, stefano.secci@lip6.fr

Abstract. *The continuous and acknowledged replication of virtual machines in different data center sites is a feasible service today. This strategy guarantees that virtual machines will have no loss of disk and memory content if a disaster occurs, at a cost of strict bandwidth and latency requirements. Considering this kind of service, in this work we propose an optimization problem to place servers in a wide area network. The goal is to guarantee that backup machines do not fail at the same time as their primary counterparts and, by using virtualization, to reduce the amount of backup servers required. The optimal results, achieved in real topologies, reduce the number of backup servers by at least 40%. Moreover, this work highlights several characteristics of the backup service according to the employed network, such as the fulfillment of latency requirements.*

Resumo. *Atualmente, a replicação contínua e confirmada de máquinas virtuais em diferentes sítios de um centro de dados é um serviço viável. Essa estratégia permite garantir perda zero do disco e do conteúdo de memória das máquinas em caso de desastres, mas possui exigências severas de banda passante e latência. Considerando esse tipo de serviço, neste trabalho propõe-se um problema de otimização para posicionar servidores em uma rede de longa distância. O objetivo é garantir que máquinas de backup não falhem ao mesmo tempo que seus correspondentes primários e, através da virtualização, reduzir o número de servidores de backup necessários. Os resultados ótimos alcançados em topologias reais reduzem o número de servidores de backup em pelo menos 40%. Além disso, este trabalho aponta diversas características do serviço em relação às redes utilizadas, como o atendimento aos requisitos de latência.*

1. Introdução

No modelo de Infraestrutura como Serviço (IaaS - *Infrastructure as a Service*), os clientes da nuvem terceirizam suas infraestruturas computacionais, executando seus serviços dentro de máquinas virtuais (VMs - *Virtual Machines*) hospedadas em um provedor. Ao utilizar serviços IaaS, um cliente abandona o controle da sua infraestrutura computacional, e confiará nesses serviços de acordo com níveis de resiliência e desempenho garantidos pelos provedores. Assim, para atrair o uso de IaaS, provedores de nuvem geralmente buscam utilizar infraestruturas resilientes de servidores e de rede [Bauer e Adams, 2012].

*Este trabalho foi realizado com recursos da FAPERJ, CNPq e CAPES.

Para tal, os provedores empregam redundância na infraestrutura, de forma a superar diversos tipos de falhas, como de *hardware* (p.ex., discos rígidos ou sistemas de refrigeração), de *software* (p.ex., programação) e humanas (p.ex., procedimentos de manutenção). Essa estratégia, entretanto, não garante a disponibilidade do serviço em eventos de desastres, que estão fora do controle do provedor. Essas situações são causadas por catástrofes naturais ou por agentes humanos não relacionadas ao provedor, que podem atingir diversos enlaces da rede além de construções inteiras de um centro de dados (DC - *Data Center*). Como as corporações estão cada vez mais migrando suas infraestruturas para a nuvem pelo paradigma IaaS, os provedores precisam estar preparados para oferecer serviços resilientes a desastres. Para tal, a nuvem IaaS deve empregar um DC com sítios geodistribuídos, de forma a eliminar pontos únicos de falha e fornecer mecanismos para realizar cópias de segurança (backups) dos serviços em execução [Couto et al., 2014a].

Este trabalho foca o projeto de DCs resilientes a desastres com perda zero de dados ou de estado das VMs. Ou seja, as VMs possuem zero RPO (*Recovery Point Objective* - Objetivo do Ponto de Recuperação) [Couto et al., 2014b]. O RPO é a diferença de tempo entre o momento da última sincronização do backup e o momento do desastre, fornecendo uma ideia da perda de dados após o desastre. Alguns serviços necessitam de um RPO baixo ou até mesmo nulo (como transações bancárias), exigindo replicação contínua dos dados. Basicamente, um serviço IaaS com zero RPO consiste em VMs que enviam continuamente cópias de backup a um servidor. Nesse caso, uma determinada operação requisitada por um usuário final só será concluída após a VM receber uma confirmação do sítio de backup de que seu estado foi corretamente replicado [Rajagopalan et al., 2012]. Como exige replicação contínua de dados, esse serviço acarreta alta utilização da banda passante da rede. Além disso, como exige confirmação dos backups, demanda baixa latência entre o sítio primário e o de backup.

Este trabalho analisa o comportamento de serviços IaaS com zero RPO em redes de longa distância (WAN - *Wide Area Network*) reais. Para tal, é proposto um esquema de posicionamento de servidores físicos em centros de dados, que aloca cada servidor primário e seu respectivo backup na rede. O posicionamento leva em consideração um modelo de falhas, de forma que um desastre não destrua um servidor primário e seu backup ao mesmo tempo. Além disso, o posicionamento proposto tira proveito da virtualização para reduzir o número de servidores de backup necessários. A ideia básica é que um servidor de backup necessita instanciar uma VM apenas após um desastre. Assim, ao invés de adotar um esquema no qual cada servidor primário possui um servidor de backup dedicado, no posicionamento proposto um único servidor de backup poderá ser compartilhado, recebendo réplicas de diferentes servidores primários. Para tanto, esses servidores primários não podem falhar ao mesmo tempo. A partir da análise das topologias WAN, mostra-se que o compartilhamento de backups permite reduzir em pelo menos 40% o número de servidores necessários. Além disso, quantifica-se a capacidade de cada uma das WANs em número de servidores primários que podem ser instalados, o que impacta diretamente o número de VMs suportadas. Requisitos mais estritos de resiliência podem reduzir em pelo menos 50% o número de servidores primários suportados.

A literatura sobre posicionamento resiliente de servidores físicos em DCs considera uma distribuição tradicional do DC, como a utilizada para redes de distribuição de conteúdo (CDN - *Content Delivery Networks*) [Habib et al., 2012, Xiao et al., 2014].

Nesses trabalhos, os serviços de um DC são replicados através de uma infraestrutura geodistribuída. Para tal, utilizam o princípio *anycast*, no qual qualquer nó que execute o serviço desejado pode responder às requisições, fazendo com que tanto os servidores primários quanto seus backups encontrem-se ativos ao mesmo tempo. Entretanto, esses trabalhos não consideram a sincronização entre as réplicas do serviço, desconsiderando requisitos de RPO. Além disso, ao considerar que os backups encontram-se ativos, não economizam o número de servidores de backup. Este trabalho difere da literatura através da consideração das réplicas do serviço, que acarreta requisitos estritos de latência e banda passante, além de prover economia de servidores de backup. Para tal, foca-se no modelo IaaS, diferente das CDNs tradicionais. Em [Couto et al., 2014a] analisou-se o compromisso entre a resiliência e latência entre os nós de um DC geodistribuído. Esse trabalho mostrou que é possível projetar um DC resiliente com baixa latência entre seus sítios. Assim, o presente trabalho foca o caso de IaaS com zero RPO e considera os requisitos de banda passante e economia de servidores. Uma área correlata ao posicionamento de servidores em um DC é o mapeamento sobrevivente de redes virtuais em redes físicas. Em [Yu et al., 2011] propõe-se um problema de posicionamento da infraestrutura virtualizada na qual recursos de backup são compartilhados, reduzindo a quantidade de recursos necessária. Por fim, outro trabalho relacionado é o posicionamento de controladores em redes definidas por software (SDN - *Software Defined Networks*). Em SDNs os elementos de comutação são gerenciados por controladores. Assim, diferentes trabalhos propõem esquemas de posicionamento de um conjunto de controladores, de forma que os elementos de comutação consigam acesso a pelo menos um deles após falhas [Muller et al., 2014].

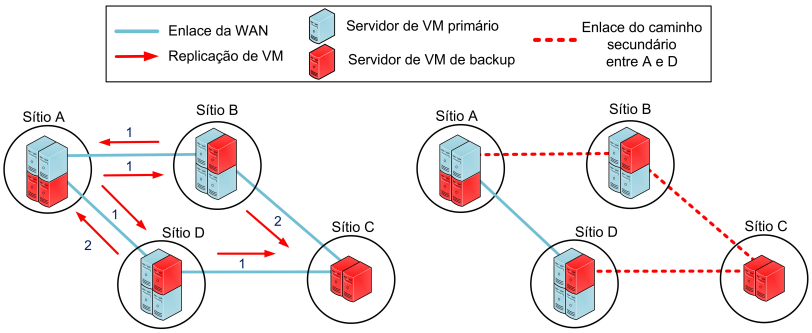
Este trabalho está organizado da seguinte forma. A Seção 2 apresenta o modelo do serviço considerado e as respectivas decisões de projeto. Com bases nessas decisões, a Seção 3 descreve o problema de otimização proposto. A Seção 4 apresenta os resultados da análise do problema proposto em redes WAN reais, enquanto a Seção 5 conclui o artigo e identifica direções futuras.

2. Modelagem e Decisões de Projeto

O problema proposto neste trabalho possui a função de distribuir servidores primários por uma WAN existente, que são utilizados para hospedar VMs de clientes IaaS. As VMs de um servidor primário são continuamente replicadas para um servidor de backup, instalado em algum outro sítio da rede. Para tal, o problema também escolhe a distribuição de servidores de backup, que hospedam as VMs dos servidores primários na ocorrência de um desastre. Para cada servidor primário, escolhe-se um servidor de backup correspondente. Note que este trabalho foca o posicionamento de servidores físicos, e não das VMs. A escolha dos servidores físicos primários que hospedam as VMs após requisições dos usuários é realizada por algoritmos de posicionamento *online* de VMs, e está fora do escopo deste trabalho [Couto et al., 2014b]. Esta seção detalha as decisões de projeto do DC, consideradas na formulação do problema de posicionamento, na Seção 3.

2.1. Replicação de VMs

O esquema de backup de VMs considerado neste trabalho é baseado na replicação contínua com reconhecimento. Um exemplo desse tipo de mecanismo é o SecondSite [Rajagopalan et al., 2012], utilizado como referência no restante deste trabalho. Para prover zero RPO, o SecondSite baseia-se em *checkpoints*. Um *checkpoint* é definido como



(a) Posicionamento dos servidores. (b) Caminho secundário entre A e D.

Figura 1. Exemplo de DC geodistribuído com replicação contínua de VMs.

o estado da VM (disco, memória, CPU) em determinado instante. Esse estado é enviado continuamente para o servidor de backup que, por sua vez, envia um reconhecimento ao seu servidor primário a cada *checkpoint* recebido. Antes do reconhecimento de um *checkpoint*, os pacotes enviados da VM ao mundo externo são acumulados em fila, não sendo enviados aos usuários finais. O envio só é permitido após o recebimento de reconhecimento positivo do *checkpoint*. O usuário final só receberá a resposta de alguma operação que realizou após essa ter sido replicada no servidor de backup. Assim, o mecanismo de replicação possui exigências severas de banda passante e latência. O alto uso de banda passante se deve à contínua replicação de dados, maior quanto mais a VM muda de estado (p.ex. quanto mais o conteúdo da memória RAM é alterado). As exigências estritas de latência se devem à necessidade de reconhecimento dos *checkpoints* antes de avançar na execução das operações. Assim, quanto menor a latência, maior será a vazão com que uma determinada VM consegue realizar suas operações.

Ao detectar uma falha no servidor primário, o SecondSite ativa a VM no servidor de backup. Para detectar a falha, emprega um terceiro tipo de servidor que realiza medições periódicas para arbitrar quando houve falhas e como o serviço deve ser recuperado. Obviamente, esses servidores também devem ser posicionados na rede de forma a detectar correta e rapidamente as falhas. Neste trabalho assume-se que esses servidores estão corretamente posicionados, conseguindo detectar todas as possíveis falhas. Esse tipo de posicionamento ainda não foi abordado na literatura, mas possui considerável semelhança com o posicionamento de controladores em SDNs [Muller et al., 2014].

2.2. Posicionamento dos Servidores

A Fig. 1(a) exemplifica o cenário considerado. Cada círculo representa um sítio posicionado em uma localidade geográfica, e os sítios são interconectados por uma WAN. Note que as VMs hospedadas em cada servidor não estão mostradas. As setas indicam que um sítio envia continuamente réplicas das VMs para seus vizinhos, e os números ao lado de cada seta indicam quantos servidores primários do sítio de origem enviam backups para o sítio de destino. O Sítio D possui três servidores primários. O backup das VMs de dois de seus servidores é enviado para o Sítio A e do terceiro servidor é enviado para o Sítio C. A figura mostra que um único sítio pode possuir servidores primários e de backup.

A Fig. 1(a) mostra que um sítio usado para backup pode atender mais backups de servidores primários do que o número de servidores de backup que possui instalados.

Por exemplo, o Sítio C possui dois servidores de backup instalados. Por outro lado, dois servidores primários do Sítio B e um do Sítio D enviam seus backups para ele, o que necessitaria de três servidores de backup no Sítio C. Entretanto, *considerando que o Sítio B e o Sítio D não estarão inacessíveis ao mesmo tempo*, o Sítio C não precisará hospedar as VMs dos três servidores primários ao mesmo tempo. Como o serviço é baseado em VMs, durante a operação normal (isto é, sem falhas) os sítios que recebem os backups não precisam manter as VMs em operação, armazenando apenas os dados referentes ao disco, conteúdo da memória e outras informações das VMs enviadas pelo servidor primário [Couto et al., 2014b, Wood et al., 2010]. Assim, em operação normal, o servidor de backup necessita apenas de capacidade de armazenamento para as VMs, fornecida por servidores de armazenamento não mostrados na figura. A capacidade de memória e processamento, fornecida pelos servidores de VMs de backup, só será necessária após o desastre, quando os procedimentos de recuperação são executados e as VMs de backup passam a operar. Assim, é possível economizar o número de servidores de VMs de backup necessários, visto que um sítio precisa ter apenas o número de servidores de backup para suportar a falha de pior caso. No exemplo do Sítio C, o pior caso constitui na falha do Sítio B, exigindo que dois de seus servidores de backup se tornem operacionais. Essa característica é considerada no problema de otimização proposto, permitindo economia significativa do número de servidores de VM necessários. É importante notar que, independente do uso desse esquema de economia, o número de servidores de armazenamento necessários para backup será sempre o mesmo. Consequentemente, desconsidera-se o posicionamento desse tipo de servidor.

Para posicionar os servidores, um requisito básico é instalar um servidor primário e seu respectivo backup em sítios diferentes e que não falhem ao mesmo tempo. Para tal, utiliza-se a Matriz de Independência de Falhas (matriz I). Cada elemento corresponde a um valor binário I_{ij}, valendo 1 se o sítio i pode se tornar inacessível ao mesmo tempo que o sítio j, 0 senão. Um sítio é considerado inacessível se, após uma falha, não possuir nenhum caminho para um *gateway* da rede ou se o próprio falhar. A matriz I é construída a partir do modelo de falhas. Neste trabalho, considera-se o modelo de grupos de risco compartilhado (SRGs - *Shared Risk Groups*) utilizado em [Couto et al., 2014a]. Um SRG é definido como um conjunto de elementos da infraestrutura suscetíveis à mesma situação de desastre. Considerando quedas de energia, um SRG pode ser o conjunto dos sítios servidos por uma mesma subestação elétrica. Assim, utilizando a noção de SRGs, o valor de I_{ij} é 1 se os sítios i e j possuírem algum SRG em comum, 0 caso contrário.

2.3. Enlace de Replicação e Caminho Secundário

Devido à severa restrição de latência imposta pela replicação de VMs, no problema formulado um determinado servidor realizará replicações de backup apenas aos vizinhos de um salto de seu sítio. Dessa forma, evita-se o aumento na latência causado por eventuais atrasos de processamento e de transmissão em roteadores intermediários. O enlace entre um sítio primário e seu respectivo backup, chamado de enlace de replicação, é utilizado para transportar as réplicas das VMs. Entretanto, caso esse enlace falhe e os dois sítios não consigam se comunicar, o processo de replicação é interrompido e as VMs do servidor primário começam a executar no modo desprotegido [Rajagopalan et al., 2012]. Nesse modo as VMs continuam operacionais, porém não realizam replicações devido à ausência de comunicação com os servidores de backup. Como em serviços com zero

RPO o tempo que a VM se encontra em modo desprotegido deve ser reduzido ao máximo, neste trabalho utiliza-se caminhos secundários entre o sítio primário e seu backup. Assim, quando for detectada uma queda no enlace de replicação, o servidor primário enviará suas replicações pelo caminho secundário. Obviamente, o caminho secundário escolhido não pode conter o enlace de replicação. A Fig. 1(b) mostra o caminho escolhido para dar continuidade ao envio de replicações entre os Sítios A e D, quando o enlace entre eles falha. Mais adiante serão analisados os compromissos de se utilizar esse caminho adicional, bem como a qualidade desse tipo de caminho em termos de latência.

3. Formulação do Problema de Otimização do Posicionamento de Servidores

Neste trabalho tenta-se maximizar o número de servidores primários e, ao mesmo tempo, minimizar a quantidade de servidores de backup adicionais necessários para abrigar as VMs após um desastre. O problema de otimização considera como parâmetros a latência e a capacidade (em Mbits/s) dos enlaces, a Matriz de Independência de Falhas, além da topologia da rede para cálculo dos caminhos secundários. A saída do problema fornece a quantidade de servidores primários e de backup a ser instalada em cada sítio, assim como o caminho secundário entre cada sítio primário e seu sítio de backup. O posicionamento proposto é realizado em duas etapas. Na primeira etapa calcula-se o caminho secundário entre cada par de sítios da rede. Na segunda etapa executa-se o problema de otimização do posicionamento de servidores físicos.

Na primeira etapa, modela-se a WAN como um grafo no NetworkX [1], no qual os nós são os sítios e cada aresta um enlace entre os sítios, com o peso correspondente à distância geográfica entre eles. Para cada par de nós vizinhos, retira-se do grafo o enlace entre esses nós e calcula-se o menor caminho entre eles utilizando o algoritmo de Dijkstra. Nessa estratégia são escolhidos os caminhos secundários com a menor latência possível, independentemente do problema de otimização subsequente. Optou-se por adotá-la de forma a forçar o menor valor de latência possível, devido às restritas exigências em relação a essa métrica. Vale lembrar que, ao fim do problema de otimização, caminhos secundários entre sítios que não replicam backup entre si não são configurados. O resultado dessa etapa consiste nos parâmetros s_{ij}^{km}, que definem os enlaces contidos no caminho secundário entre cada par de sítios k e m. Ou seja, para um enlace entre i e j, tem-se $s_{ij}^{km} = 1$ se esse enlace está no caminho secundário entre k e m, ou $s_{ij}^{km} = 0$ senão.

Na segunda etapa, utiliza-se a ferramenta IBM ILOG CPLEX 12.5.1 para executar o problema de programação linear inteira (ILP - *Integer Linear Programming*) formulado a seguir. Esse problema escolhe o posicionamento de cada servidor físico primário e seu respectivo backup, considerando os objetivos da otimização e restrições mencionados anteriormente. A Tabela 1 lista as notações utilizadas e seus tipos. Notações do tipo conjunto e parâmetro se referem aos dados do problema, enquanto as variáveis são ajustadas pelo algoritmo de otimização. A formulação ILP é a seguinte:

$$\text{maximizar} \quad \sum_{i \in \mathcal{D}} (x_i - b_i) \tag{1}$$

$$\text{sujeito a} \quad c_{ij} \cdot I_{ij} = 0 \quad \forall i, j \in \mathcal{D}. \tag{2}$$

[1] A ferramenta NetworkX está disponível em `http://networkx.github.io/`

Tabela 1. Notações utilizadas no problema.

Notação	Descrição	Tipo
$\mathcal{D}$	Sítios candidatos	Conjunto
I_{ij}	Valor binário indicando se o sítio i pode se tornar inacessível ao mesmo tempo que o sítio j	Parâmetro
Δ_{ij}	Atraso de propagação (latência) do enlace entre os sítios i e j	Parâmetro
W_{ij}	Capacidade do enlace entre os sítios i e j	Parâmetro
s_{ij}^{km}	Valor binário indicando se o enlace entre i e j pertence ao caminho secundário entre k e m	Parâmetro
α	Fração máxima da capacidade dos enlaces a ser utilizada pela replicação	Parâmetro
γ	Parâmetro binário indicando se existirá um caminho secundário entre sítios de backup e primários	Parâmetro
B	Consumo de banda, em Mbits/s, gerado pela transferência contínua do backup de um servidor primário	Parâmetro
L_{worst}	Máxima latência permitida entre um servidor primário e seu backup	Parâmetro
U_{max}	Máximo número de sítios utilizados	Parâmetro
x_i	Número de servidores primários na localização i	Variável
b_i	Número de servidores de backup na localização i	Variável
u_i	Valor binário indicando se o sítio i está ativo ($(x_i + b_i) > 0$)	Variável
r_{ij}	Quantidade de banda disponível para replicações do sítio i para o sítio j	Variável
c_{ij}	Número de backups do sítio i no sítio j	Variável
e_{ij}	Valor binário indicando se o sítio i possui backups no sítio j ($c_{ij} > 0$)	Variável
y_{kij}	Valor binário indicando se o sítio k abriga backups dos sítios i **e** j ($e_{ik} = 1$ e $e_{jk} = 1$)	Variável

$$\sum_{j \in \mathcal{D}} c_{ij} = x_i \quad \forall i \in \mathcal{D}. \tag{3}$$

$$M \cdot e_{ij} - c_{ij} \geq 0 \quad \forall i, j \in \mathcal{D}. \tag{4}$$

$$e_{ij} \leq c_{ij} \quad \forall i, j \in \mathcal{D}. \tag{5}$$

$$y_{kij} \geq e_{ik} + e_{jk} - 1 \quad \forall k, i, j \in \mathcal{D},\ i < j. \tag{6}$$

$$y_{kij} \leq e_{ik} \quad \forall k, i, j \in \mathcal{D},\ i < j. \tag{7}$$

$$y_{kij} \leq e_{jk} \quad \forall k, i, j \in \mathcal{D},\ i < j. \tag{8}$$

$$\sum_{i,j \in \mathcal{D},\ i<j} (I_{ij} \cdot y_{kij}) = 0 \quad \forall k \in \mathcal{D}. \tag{9}$$

$$b_j - c_{ij} \geq 0 \quad \forall i, j \in \mathcal{D}. \tag{10}$$

$$B \cdot c_{ij} \leq r_{ij} \quad \forall i, j \in \mathcal{D}. \tag{11}$$

$$r_{ij} \leq \alpha \cdot W_{ij} - \gamma \cdot B \cdot c_{km} \cdot s_{ij}^{km} \quad \forall i, j \in \mathcal{D} \quad \forall k, m \in \mathcal{D}. \tag{12}$$

$$e_{ij} \cdot \Delta_{ij} \leq L_{worst} \quad \forall i, j \in \mathcal{D}. \tag{13}$$

$$M \cdot u_i - (x_i + b_i) \geq 0 \quad \forall i \in \mathcal{D}. \tag{14}$$

$$u_i \leq (x_i + b_i) \quad \forall i \in \mathcal{D}. \tag{15}$$

$$\sum_{i \in \mathcal{D}} u_i \leq U_{max}. \tag{16}$$

$$x_i \geq 0, \quad b_i \geq 0,\ \forall\, i \in \mathcal{D}; c_{ij} \geq 0\ \forall\, i, j \in \mathcal{D}. \tag{17}$$

$$x_i \in \mathbb{Z}, \quad b_i \in \mathbb{Z}\ \forall\, i \in \mathcal{D}; \quad r_{ij} \in \mathbb{Z},\ c_{ij} \in \mathbb{Z}\ \forall\, i, j \in \mathcal{D}; u_i \in \{0, 1\},\ \forall\, i \in \mathcal{D}; \quad e_{ij} \in \{0, 1\}\ \forall\, i, j \in \mathcal{D}; y_{kij} \in \{0, 1\}\ \forall\, k, i, j \in \mathcal{D}. \tag{18}$$

O objetivo do problema, dado pela Eq. (1), tenta encontrar o melhor compromisso entre o número de servidores primários ($\sum_{i \in \mathcal{D}} x_i$) e o número de servidores utilizados

para backup ($\sum_{i \in \mathcal{D}} b_i$). Para cada servidor instalado em um sítio, o problema tentará reduzir em uma unidade o número de servidores de backup. Dessa forma, a função objetivo pode ser vista como a economia, em número de servidores, dos servidores de backup necessários para instalar o serviço. No pior caso a função objetivo será zero, enquanto no melhor caso se aproximará do número de servidores primários instalados.

A Eq. (2) força os servidores primários de um sítio i a só possuírem backup em um sítio j (ou seja, $c_{ij} > 0$) se i e j não puderem se tornar inacessíveis ao mesmo tempo ($I_{ij} = 0$). A Eq. (3) define que o número de backups que um sítio i deverá espalhar pela rede seja igual ao número de servidores primários desse sítio.

As Eqs. (4) e (5) são utilizadas para calcular as variáveis binárias e_{ij}, que recebem valor 1 se $c_{ij} > 0$ e 0 caso contrário. O parâmetro M da Eq. (4) é apenas um valor grande o suficiente para ser sempre maior ou igual a qualquer possível valor das variáveis c_{ij} dessa equação e das somas $x_i + u_i$ da Eq. (14), mostrada adiante. Adota-se, de forma conservadora, $M = 1 \times 10^9$. As Eqs. (6), (7) e (8), utilizadas para calcular as variáveis binárias y_{kij}, são equivalentes à operação lógica E entre e_{ik} e e_{jk}.

A Eq. (9) é responsável pela economia de servidores de backup. Essa equação garante que se dois sítios i e j podem se tornar inacessíveis ao mesmo tempo (ou seja, $I_{ij} = 1$), eles não poderão hospedar seus backups em um mesmo sítio k. Em outras palavras, se $I_{ij} = 1$, i e j não poderão compartilhar servidores de backup. Note que a Eq. (9) influencia as variáveis y_{kij} e, consequentemente, as variáveis e_{ij} e c_{ij}, que definem o posicionamento dos backups. Como o problema garante que sítios que podem se tornar inacessíveis ao mesmo tempo não compartilham backups, o número de servidores de backup pode ser calculado utilizando a Eq. (10), que é equivalente a fazer $b_j = \max_{j \in \mathcal{D}}(c_{ij})$. Isto é, o número de servidores de backup a ser instalado em um sítio j é dado pelo máximo número de backups que qualquer outro sítio da rede envia para ele. Voltando à Fig. 1(a), o Sítio C recebe dois backups do Sítio B e um backup do Sítio D. Assim, o Sítio C possui dois servidores de backup para suportar a falha do Sítio B, que consiste no sítio que o envia o maior número de backups.

As Eqs. (11) e (12) tratam das restrições de banda passante. Na Eq. (11), $B \cdot c_{ij}$ representa a quantidade de banda necessária para a replicação de c_{ij} servidores primários de i em j. Note que, para cada backup, uma quantidade de banda de B Mbits/s é utilizada continuamente no enlace entre os dois sítios. Essa equação especifica que o consumo de banda em um determinado enlace não pode ultrapassar o valor definido pela variável r_{ij}, dada pela Eq. (12). Essa equação define a quantidade de banda disponível r_{ij} para realizar replicações entre os sítios i e j, considerando a banda reservada aos caminhos secundários que passam pelo enlace entre i e j. Nessa equação, $\alpha \cdot W_{ij}$ representa a quantidade de banda do enlace alocada para todo o serviço de IaaS com backup contínuo. O parâmetro W_{ij} representa a capacidade do enlace, zero caso os sítios i e j não possuam um enlace entre si. Dessa forma, apenas sítios vizinhos de um salto poderão replicar backups entre si. O termo $\gamma \cdot B \cdot c_{km} \cdot s_{ij}^{km}$ é a quantidade de banda utilizada por um determinado caminho secundário entre dois sítios k e m, que passa pelo enlace entre i e j. Note que, nesse problema, a Eq. (12) é equivalente a fazer $r_{ij} = \min_{k,m \in \mathcal{D}}(\alpha W_{ij} - B \cdot c_{km} \cdot s_{ij}^{km})$. Isto é, de todos os caminhos secundários que passam por um determinado enlace, contabiliza-se em r_{ij} apenas aquele que consome a maior quantidade de banda. Isso é possível assumindo que nenhum enlace da rede falhe ao mesmo tempo e, dessa forma, apenas um caminho

secundário poderá estar ativo em todo o DC. Assim, em um determinado enlace, é necessário apenas reservar banda necessária para o pior caso de caminho secundário. Em outras palavras, os caminhos secundários são provisionados no esquema de proteção compartilhada [Ramamurthy et al., 2003]. O parâmetro γ da Eq. (12) é utilizado para desabilitar a utilização de caminhos secundários, liberando recursos do enlace para replicações. Dessa forma, se $\gamma = 0$ tem-se simplesmente $r_{ij} = \alpha \cdot W_{ij}$.

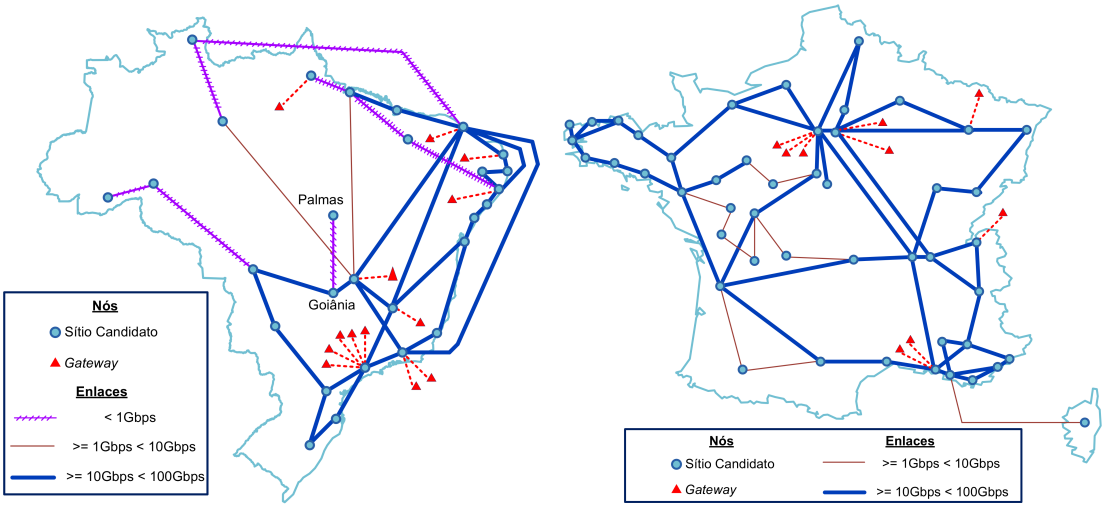
A Eq. (13) define os requisitos de latência da replicação contínua. Um sítio i só poderá replicar backups para o sítio j (ou seja, $e_{ij} = 1$) se o enlace entre eles possuir latência Δ_{ij} inferior ou igual à máxima latência permitida (L_{worst}). As Eqs. (14), (15) e (16) limitam o número de sítios escolhidos para instalar servidores primários ou de backup, com base no número máximo de sítios, U_{max}. Finalmente, as Eqs. (17) e (18) definem, respectivamente, o limitante inferior e o domínio de cada variável.

Alguns requisitos comuns em serviços IaaS não são considerados no problema, como a proximidade dos servidores primários com os usuários finais. Desconsideram-se esses requisitos neste trabalho para permitir uma melhor análise de um serviço com zero RPO, cujos principais requisitos são a independência das falhas entre servidores primários e seus respectivos backups, além da baixa latência de replicação entre eles. Como visto anteriormente, a baixa latência é importante, pois influencia diretamente o tempo de resposta das aplicações executando nas VMs dos servidores primários.

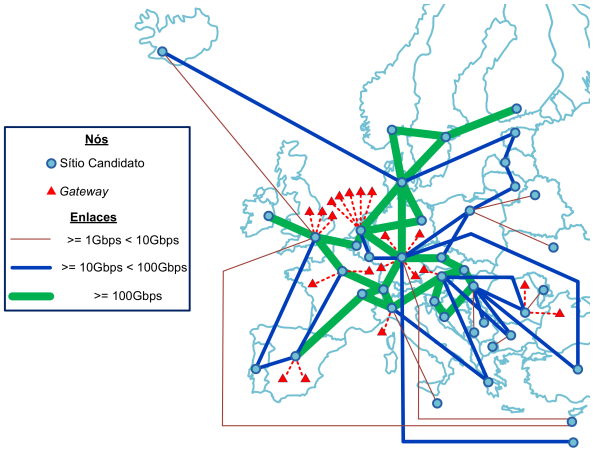
4. Avaliação

O problema de otimização formulado na Seção 3 é utilizado neste trabalho para posicionar servidores em WANs reais de educação e pesquisa. Essas redes são formadas por diversos pontos de presença (PoP - *Point of Presence*), que são os possíveis sítios do DC no contexto deste trabalho. Assim, são adotadas as topologias da RNP (Fig. 2(a)) no Brasil, RENATER (Fig. 2(b)) na França, e GEANT (Fig. 2(c)) na Europa. Cada figura das redes utilizadas mostra seus sítios e *gateways* de acesso à Internet.

A partir das topologias mencionadas, calcula-se os parâmetros de entrada do problema de otimização. O valor de latência Δ_{ij} de cada enlace é dado pelo atraso de propagação entre os sítios i e j. Considera-se que a rede é bem provisionada e, assim, os atrasos de fila e de transmissão são desprezíveis. O atraso de propagação é diretamente proporcional à distância entre os dois sítios, que é estimada neste trabalho como o comprimento de uma linha reta entre os centros das duas cidades que hospedam os sítios. Para o cálculo desse atraso utiliza-se uma velocidade de propagação de 2×10^8 m/s, comumente adotada para análise de redes ópticas [Couto et al., 2014a]. A Matriz de Independência de Falhas (matriz I) é calculada a partir do modelo de SRGs de falha única, adotado também em [Couto et al., 2014a]. Nesse modelo, apenas um enlace ou sítio da rede falha por vez. Note que, apesar do modelo de falha única, dois sítios podem se tornar inacessíveis ao mesmo tempo. Por exemplo, na rede da Fig. 2(a), se o sítio de Goiânia falhar, o sítio de Palmas também ficará inacessível. A capacidade W_{ij} de cada enlace é mostrada na Fig. 2. Esses valores foram obtidos no *website* oficial de cada uma das redes. A rede é modelada por um grafo direcionado, ou seja, a capacidade mostrada na figura corresponde à banda passante disponível em cada sentido do enlace entre dois sítios. Apesar de o grafo ser direcionado, considera-se no modelo de falhas que a ruptura de um determinado enlace representa a falha nos dois sentidos desse enlace.



(a) RNP, 28 sítios conectados por 38 enlaces. (b) RENATER , 48 sítios conectados por 67 enlaces.



(c) GEANT, 41 sítios conectados por 58 enlaces.

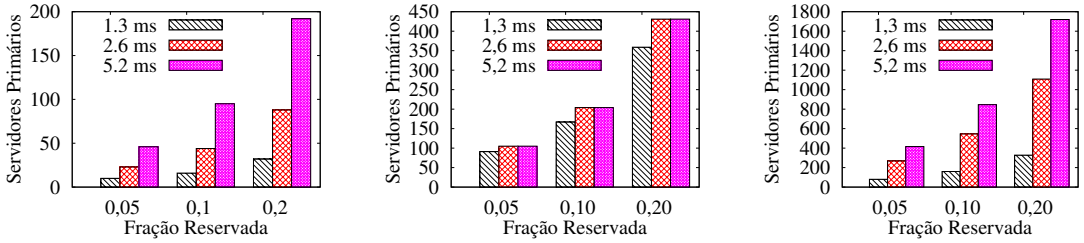
Figura 2. Topologias de redes de ensino e pesquisa consideradas na avaliação.

O consumo de banda gerado pela replicação contínua de cada servidor operacional, dado pelo parâmetro B, é fixado em 240 Mbits/s. Esse valor foi retirado do trabalho no qual o SecondSite foi proposto [Rajagopalan et al., 2012], e corresponde aproximadamente ao consumo de banda ao replicar um servidor primário que hospeda quatro VMs, sendo duas executando um *benchmark* de servidor web e duas executando um *benchmark* de banco de dados. O valor de B escolhido neste trabalho é usado apenas como referência, podendo variar consideravelmente dependendo da aplicação em execução nas VMs. Assim, no projeto de DCs, B deverá ser especificado em função de um Acordo de Nível de Serviço (SLA - *Service Level Agreement*), no qual o provedor garantirá uma certa banda dedicada ao backup contínuo. Além disso, cada conjunto de servidores primários pode possuir um requisito diferente de banda de replicação, o que tornaria necessária a utilização de diferentes valores de B. O problema proposto pode ser facilmente modificado para considerar diferentes valores de B. Entretanto, optou-se por utilizar apenas um único valor de B para simplificar a análise. Outro parâmetro que foi fixado para todas as avaliações desta seção é o máximo número de sítios utilizados, dado por U_{max}. Fixou-se U_{max} no número total de sítios existentes, para não limitar o número de sítios utilizados.

Finalmente, salvo indicação contrária, o parâmetro γ é fixado em 1 na análise a seguir. Isso significa que o problema considera a utilização de caminhos secundários.

4.1. Capacidade do Serviço e Economia

A partir do problema proposto e dos parâmetros escolhidos anteriormente, analisa-se a capacidade do serviço de IaaS resiliente nas três redes consideradas. A capacidade do serviço, medida em número de servidores primários suportados, é analisada para diferentes valores de banda disponível (α) e diferentes valores de latência máxima tolerada (L_{worst}). A Fig. 3 apresenta os resultados para cada rede, na qual cada gráfico agrupa no eixo X diferentes valores de α e cada barra representa um L_{worst} diferente. O L_{worst} mais baixo (isto é, 1,3 ms) foi escolhido com base nos experimentos realizados no artigo do SecondSite [Rajagopalan et al., 2012], nos quais replica-se um servidor em um enlace de 260 km entre as cidades canadenses de Vancouver e Kamloop. Essa distância acarreta uma latência de 1,3 ms, considerando a velocidade de propagação utilizada neste trabalho. Os demais valores utilizados são relaxamentos dos requisitos de latência, representando o dobro e o quádruplo desse valor de referência. Os resultados da Fig. 3 mostram, como esperado, que quanto maior a banda reservada e quanto mais relaxado o requisito de latência, maior o número de servidores primários suportados. Note que na rede RENATER o número de servidores instalados não se altera ao relaxar o requisito de latência de 2,6 ms para 5,2 ms. Isso ocorre pois essa rede abrange a França metropolitana, uma área significativamente menor que a área coberta pelas outras redes. Assim, a maioria de seus enlaces já atende aos requisitos de latência quando $L_{worst} = 2,6$ms.



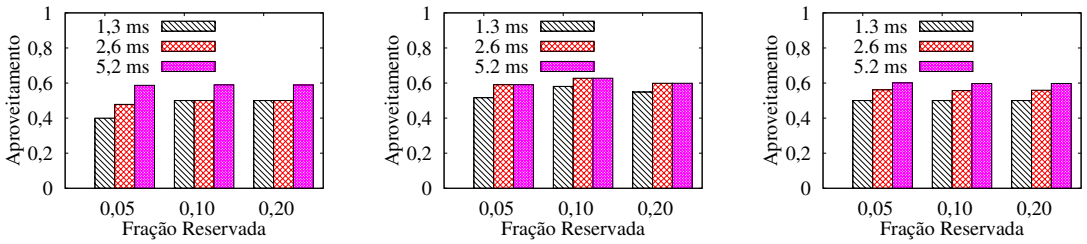
(a) RNP. (b) RENATER. (c) GEANT.

Figura 3. Número de Servidores Primários.

A Fig. 4 mostra o resultado da economia de servidores de backup alcançada pela otimização do posicionamento. A economia é quantificada pela métrica Aproveitamento de Servidores (AS), definida como a relação entre o número de servidores de backup que não foram necessários instalar e o número total de servidores primários, dada por:

$$\text{AS} = \frac{\sum_{i \in \mathcal{D}}(x_i - b_i)}{\sum_{i \in \mathcal{D}}(x_i)} = 1 - \frac{\sum_{i \in \mathcal{D}}(b_i)}{\sum_{i \in \mathcal{D}}(x_i)}. \tag{19}$$

Assim, AS varia no intervalo $[0, 1[$, e quanto maior seu valor, maior a economia. A Fig. 4 mostra que, para todas as redes e todos os valores de α e L_{worst} avaliados, o aproveitamento é igual ou maior que 40%. Isso mostra que o posicionamento proposto possibilita uma economia significativa de servidores de backup nas WANs consideradas. Além disso, os resultados mostram que relaxar o requisito de latência pode melhorar a economia de servidores, visto que mais opções de posicionamento estarão disponíveis.



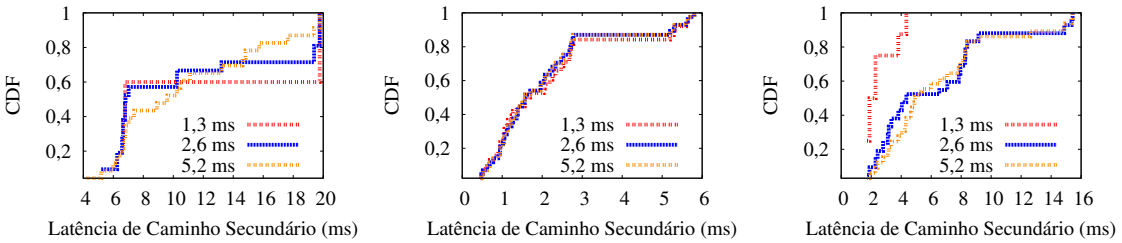
(a) RNP. (b) RENATER. (c) GEANT.

Figura 4. Economia de servidores físicos.

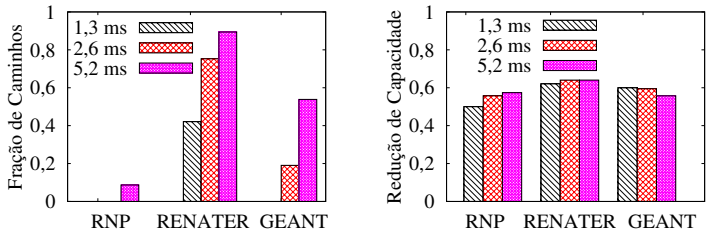
4.2. Caminhos Secundários

Como já visto, o posicionamento deste trabalho restringe o valor máximo de latência no enlace entre dois sítios que replicam backups entre si. Entretanto, o problema não restringe esse valor para os caminhos secundários. Isto significa que, caso o enlace de replicação entre dois sítios falhe, esses realizarão suas replicações entre si utilizando um caminho que poderá não atender aos requisitos de latência definidos por L_{worst}. Dessa forma, o tempo de resposta das aplicações que as VMs executam poderá aumentar. A Fig. 5 mostra características dos caminhos secundários, considerando apenas os caminhos entre sítios que replicam backups entre si. Essas características são extraídas da saída do problema de otimização de cada rede. Os resultados são apresentados para $\alpha = 0,05$, mas os valores alcançados para os outros valores de α considerados são muito próximos aos da Fig. 5. As Figs. 5(a), 5(b) e 5(c) apresentam a Função de Distribuição Cumulativa (CDF - *Cumulative Distribution Function*) da latência dos caminhos secundários para cada uma das redes utilizadas. Note que em todas as redes existem diversos caminhos secundários com valores de latência bem mais altos do que os valores L_{worst} especificados. Na RNP, alguns caminhos possuem valores de latência próximos a 20 ms. Para uma melhor visualização dessa característica, a Fig. 5(d) apresenta a fração de caminhos secundários que atendem ao requisito de latência L_{worst}. O eixo X agrupa os resultados obtidos com diferentes valores de L_{worst} para cada rede analisada. Note que o requisito mais estrito, de 1,3 ms, é atendido apenas por 40% dos caminhos na RENATER e por nenhum caminho da RNP e GEANT. Esse melhor desempenho da RENATER ocorre pois essa rede abrange uma área significativamente menor que as outras, como mencionado anteriormente. Para o requisito mais relaxado, de 5,6 ms, a RENATER é capaz de atender ao requisito L_{worst} em aproximadamente 90% dos caminhos. Apesar desse resultado favorável, as demais redes, e mesmo a própria RENATER para requisitos mais estritos, apresentam caminhos secundários com altos valores de latência. Isso mostra que apenas o posicionamento proposto não é suficiente para garantir baixos valores de latência nos caminhos secundários, e que a topologia da WAN deve ser modificada para oferecer tal serviço. Esse projeto da WAN é importante, sobretudo em redes que abrangem grandes áreas geográficas. Este trabalho foca o posicionamento de servidores, considerando que a WAN já está instalada. O alinhamento do posicionamento de servidores ao projeto da WAN, recomendado em [Couto et al., 2014b], é um tema de trabalho futuro.

Como os possíveis caminhos secundários da rede podem apresentar alta latência, o projetista do DC pode escolher não configurar esses caminhos, fazendo $\gamma = 0$ no problema formulado na Seção 3. Dessa forma, mais banda passante estará disponível para a



(a) CDF da latência na RNP. (b) CDF da latência na RENATER. (c) CDF da latência na GEANT.



(d) Atendimento de L_{worst}. (e) Sobrecarga.

Figura 5. Características dos caminhos secundários ($\alpha = 0,05$).

instalação de servidores primários e seus respectivos backups. Por outro lado, se houver falha do enlace entre dois sítios que replicam entre si, os servidores primários correspondentes deverão ser pausados para evitar a execução de operações não replicadas. Essa estratégia, entretanto, diminui a disponibilidade dos servidores primários (isto é, a fração de tempo que permanecem operacionais). Outra estratégia é deixar o servidor primário em modo desprotegido (isto é, operacional, mas sem realizar backup), afetando os valores de RPO em prol da disponibilidade.

Para analisar o quanto a utilização de caminhos secundários reduz a capacidade de instalação de servidores primários, executou-se o problema com os mesmos parâmetros das seções anteriores, mas fazendo $\gamma = 0$. Denotando por S e S', respectivamente, o número de servidores primários suportados em uma rede quando utilizam-se caminhos secundários ($\gamma = 1$) e quando esses não são utilizados ($\gamma = 0$), calcula-se a redução da capacidade pela expressão $1 - \frac{S}{S'}$. A Fig. 5(e) mostra, para as diferentes redes e requisitos de latência, o valor da redução quando a fração máxima da banda utilizada nos enlaces (isto é, α) é de 0,05. Os resultados mostram que os caminhos secundários reduzem significativamente a capacidade do serviço, sendo a redução superior a 50% para todos os resultados da Fig. 5(e). Os resultados para os outros valores de α, omitidos por questões de espaço, apresentam o mesmo comportamento, sendo sempre superiores a 50%. Assim, essa análise mostra que o projetista do DC pode aumentar significativamente a capacidade do serviço se escolher não utilizar caminhos secundários nas topologias consideradas.

5. Conclusões e Trabalhos Futuros

Neste trabalho foi proposto o posicionamento de servidores físicos em DCs geodistribuídos, através da formulação de um problema de otimização com objetivo de suportar serviço IaaS com zero RPO. Utilizando essa proposta, mostrou-se ser possível reduzir significativamente o número de servidores necessários, através do compartilhamento de servidores de backups. Os resultados para todas as redes consideradas obtiveram no mínimo

40% de economia no número de servidores. Tal economia pode ser ainda maior se os requisitos de latência forem relaxados. Este trabalho também apresentou as características do DC geodistribuído se forem configurados caminhos secundários entre sítios que realizam réplicas entre si. Esses caminhos são utilizados caso o enlace entre dois sítios falhe. Os resultados mostraram que os caminhos secundários podem apresentar alta latência, desrespeitando os requisitos do serviço de replicação. Por fim, mostra-se que a existência de caminhos secundários adiciona alta sobrecarga à rede, devido à quantidade necessária de banda reservada. Os resultados mostraram que, em todas as redes consideradas, seria possível adicionar pelo menos o dobro do número de servidores primários caso os caminhos secundários não fossem configurados.

Como trabalhos futuros, pretende-se propor um algoritmo de projeto da WAN que atue em conjunto com o posicionamento de servidores para, por exemplo, oferecer caminhos secundários com melhor qualidade. Outra direção promissora é considerar o posicionamento dos servidores que detectam falhas de forma a otimizar o tempo de recuperação das VMs e permitir uma detecção mais acurada.

Referências

Bauer, E. e Adams, R. (2012). *Reliability and availability of cloud computing*. John Wiley & Sons.

Couto, R. S., Secci, S., Campista, M. E. M. e Costa, L. H. M. K. (2014a). Latência versus sobrevivência no projeto de centros de dados geograficamente distribuídos. Em *XXXII SBRC*, p. 809—822.

Couto, R. S., Secci, S., Campista, M. E. M. e Costa, L. H. M. K. (2014b). Network design requirements for disaster resilience in IaaS clouds. *IEEE Communications Magazine*, 52(10):52–58.

Habib, M. F., Tornatore, M., De Leenheer, M., Dikbiyik, F. e Mukherjee, B. (2012). Design of disaster-resilient optical datacenter networks. *Journal of Lightwave Technology*, 30(16):2563–2573.

Muller, L. F., Oliveira, R. R., Luizelli, M. C., Gaspary, L. P. e Barcellos, M. P. (2014). Survivor: an enhanced controller placement strategy for improving SDN survivability. Em *IEEE GLOBECOM*, p. 1944—1950.

Rajagopalan, S., Cully, B., O'Connor, R. e Warfield, A. (2012). SecondSite: disaster tolerance as a service. Em *ACM Virtual Execution Environments (VEE)*, p. 97–108.

Ramamurthy, S., Sahasrabuddhe, L. e Mukherjee, B. (2003). Survivable WDM mesh networks. *Journal of Lightwave Technology*, 21(4):870.

Wood, T., Cecchet, E., Ramakrishnan, K., Shenoy, P., Van der Merwe, J. e Venkataramani, A. (2010). Disaster recovery as a cloud service: Economic benefits & deployment challenges. Em *2nd USENIX Workshop on Hot Topics in Cloud Computing*.

Xiao, J., Wen, H., Wu, B., Jiang, X., Ho, P.-H. e Zhang, L. (2014). Joint design on DCN placement and survivable cloud service provision over all-optical mesh networks. *IEEE Transactions on Communications*, 62(1):235–245.

Yu, H., Anand, V., Qiao, C. e Sun, G. (2011). Cost efficient design of survivable virtual infrastructure to recover from facility node failures. Em *IEEE ICC*, p. 1–6.

Uma Avaliação da Influência da Velocidade dos Nós no Estabelecimento de Caminhos em Redes Ad Hoc Veiculares

Dayro A. B. Hernandez, Dianne S. V. Medeiros,
Miguel Elias M. Campista e Aloysio de Castro P. Pedroza

[1]GTA / PEE-COPPE / DEL-Poli
Universidade Federal do Rio de Janeiro (UFRJ)

{dayro, dianne, miguel, aloysio}@gta.ufrj.br

Abstract. *Contacts exploitation in mobile wireless networks depends on the knowledge of node mobility patterns. This assumption seems trivial, but many communication protocols admit the existence of multihop paths, disregarding how often this event indeed occurs. In this direction, this paper evaluates the influence of nodes speed on path establishment, whether they are single or multihop paths. The objective is to evaluate how much of communication protocols overload could be reduced, if node speed was considered. Results are obtained through analysis of vehicular network traces, both real and synthetic, and via a case study simulation. While the analysis reveals that contacts are concentrated at a single hop already from relatively low speeds, the simulation shows that flooding procedures aiming path discovery tend to be not so efficient in mobile wireless networks with high mobility, in despite of theoretically forming optimal paths between nodes.*

Resumo. *O aproveitamento dos contatos em redes móveis sem-fio depende do conhecimento dos padrões de mobilidade dos nós. Apesar de essa premissa parecer trivial, muitos protocolos de comunicação assumem que caminhos compostos por múltiplos saltos podem existir, ignorando a frequência com a qual isso de fato ocorre. Nessa direção, este trabalho avalia a influência da velocidade relativa dos nós no estabelecimento de caminhos, sejam eles de um único ou de múltiplos saltos. O objetivo é avaliar o quanto a sobrecarga dos protocolos de comunicação poderia ser reduzida, caso a velocidade dos nós fosse levada em conta. Os resultados são obtidos a partir de análises sobre traços reais e sintéticos de redes veiculares e a partir de simulação de um estudo de caso. Enquanto a análise revela que os contatos se concentram a apenas um salto já a partir de velocidades relativamente baixas, as simulações mostram que procedimentos de inundação para descoberta de caminhos tendem a ser pouco eficientes em redes móveis com elevada mobilidade, apesar de teoricamente formarem caminhos ótimos entre os nós.*

1. Introdução

A caracterização dos padrões de mobilidade das redes móveis ad hoc permanece um desafio para a pesquisa, principalmente em ambientes mais recentes como o de redes veiculares [Olariu et al., 2010]. Devido à alta movimentação dos nós e à ausência de infraestrutura previamente instalada, as redes formadas nesses ambientes apresentam

alguns problemas complexos, como conectividade intermitente e ausência de caminhos fim-a-fim. Esses problemas são agravados pelos desafios do próprio meio sem-fio que pode introduzir altas taxas de erro, impedindo que aplicações com requisitos mínimos de qualidade de serviço sejam atendidas [Zhao et al., 2012].

Os padrões de mobilidade em redes móveis ad hoc e as soluções para superar os desafios vêm sendo investigados na literatura. Rezende et al. apresenta um protocolo de previsão de vizinhança capaz de antecipar a disponibilidade dos enlaces [Rezende et al., 2009]. Outros trabalhos contribuem com a análise da frequência dos contatos entre os nós móveis [Conan et al., 2007, Gonzalez et al., 2008, Passarella e Conti, 2011], considerando que dois nós estão em contato quando têm alcance mútuo e estão em intercontato quando se encontram fora do alcance direto. Trabalhos mais recentes [Phe-Neau et al., 2012, Phe-Neau et al., 2013] estendem a visão de vizinhança, incluindo os nós que podem ser alcançados com alguns saltos de distância. Essa visão estendida permite obter padrões de movimentação da vizinhança e alcançar um maior aproveitamento dos contatos em redes móveis. Nenhum desses trabalhos, porém, considera a velocidade relativa entre os nós ao investigar o comportamento da vizinhança, seja através da visão tradicional ou da estendida. A velocidade é um parâmetro que introduz mais um grau de complexidade às topologias, uma vez que modifica o panorama dos tempos de contato entre os nós e, consequentemente, o aproveitamento da vizinhança.

O objetivo deste trabalho é avaliar como os contatos ocorrem em cenários altamente dinâmicos, como as redes ad hoc veiculares, e como se comportam os caminhos a mais de um salto de distância. As redes veiculares são utilizadas devido ao intervalo mais amplo de velocidades, seja para um único veículo ou para vários entre si. Os resultados obtidos através da análise de conjuntos de dados reais e sintéticos permitem identificar os ambientes em que é mais propício usar comunicações através de múltiplos saltos, definindo até que ponto a vizinhança a mais de um salto pode ser considerada viável. Nota-se que os contatos se concentram a apenas um salto já a partir de velocidades relativamente baixas, o que pode tornar as tentativas de comunicações a múltiplos saltos ineficientes. Essa última afirmação é ainda investigada através de um estudo de caso simulado, no qual se procura reduzir a sobrecarga de controle da rede através da diminuição do alcance das mensagens de controle. A ideia é mostrar que procedimentos que contam com a existência de caminhos de múltiplos saltos, p.ex. procedimentos de inundação, não trazem benefícios para as redes veiculares atuais. Essa constatação permite concluir que o aproveitamento de contatos de forma oportunística é a alternativa com maior potencial nos cenários móveis altamente dinâmicos e, portanto, devem ser exploradas ao máximo.

Este artigo está organizado da seguinte forma. A Seção 2 apresenta definições relacionadas à vizinhança de um nó. A Seção 3 define o problema a ser estudado. A Seção 4 mostra a proposta da análise de vizinhança realizada. Já a Seção 5 descreve as caraterísticas dos traços utilizados. A Seção 6 apresenta os resultados da análise de vizinhança proposta. A Seção 7 apresenta o estudo de caso simulado e os resultados obtidos. Por fim, a Seção 8 conclui este trabalho e apresenta as direções futuras.

2. Definições

A análise proposta neste artigo requer conhecimento sobre conceitos relacionados à vizinhança de um nó. Essas definições, inicialmente propostas

em [Phe-Neau et al., 2013], são apresentadas a seguir.

Definição 1. k-vizinhança: *A k-vizinhança de um nó i é o conjunto de todos os nós cujo caminho mais curto a partir de i é de no máximo k saltos [Phe-Neau et al., 2013].*

Em outras palavras, k é o número máximo de saltos necessários para que um nó estabeleça contato com outro nó em sua k-vizinhança em uma rede móvel sem-fio. A Figura 1(a) ilustra os contatos de múltiplos saltos estabelecidos pelo nó i até sua 3-vizinhança. Observa-se que existe contato direto entre o nó i e os nós mais próximos a ele (1-vizinhança). Entretanto, com o aumento do limite da k-vizinhança, torna-se possível o contato entre nós que não estão em contato direto através de nós intermediários.

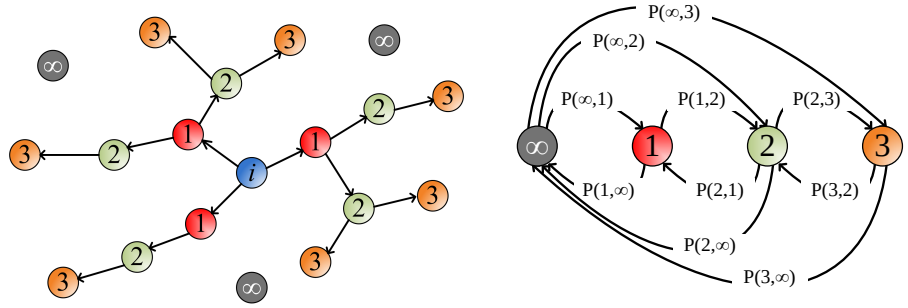
Definição 2. Estado: *Entende-se por Estado k a distância entre um par de nós medida em número de saltos.*

Assim, dois nós que estão em contato direto se encontram no Estado 1. Se for exigido um salto adicional através de um nó intermediário para estabelecer o caminho de comunicação, diz-se que os nós estão no Estado 2, e assim por diante. Se não existir caminho entre um par de nós, o estado é considerado infinito [Phe-Neau et al., 2013].

Em razão da dinâmica das topologias móveis, a mudança de estado entre os pares de nós é frequente. Como consequência, existe uma probabilidade de transição entre estados, que consiste na probabilidade de um par de nós estar em um estado diferente no próximo instante de tempo. Tal probabilidade captura a mudança na distância entre o par de nós, que pode ser diferente em instantes de tempo consecutivos. Existe também a probabilidade de dois nós fora de contato (Estado infinito) entrarem em contato, ou de dois nós em contato se encontrarem posteriormente em Estado infinito. A Figura 1(b) ilustra as mudanças de estado de um nó com as respectivas probabilidades de transição.

Definição 3. Linha de tempo da vizinhança: *Consiste na progressão da menor distância entre quaisquer dois nós ao longo do tempo. Através do uso dessas linhas é possível realizar diferentes análises probabilísticas [Phe-Neau et al., 2013].*

As linhas de tempo da vizinhança são geradas a partir dos traços de contato entre nós em um cenário móvel e armazenam a menor distância entre pares de nós ao longo do



(a) Vizinhança do nó i. (b) Probabilidades de transição.

Figura 1. (a) O conjunto de nós em contato direto é sua 1-vizinhança. Os nós alcançáveis com dois saltos são a 2-vizinhança e assim por diante. O conjunto de nós fora do alcance de i têm distância infinita. (b) Quando um par de nós se encontra no Estado 3 é possível que mude para os Estados 1, 2 ou infinito no próximo instante de tempo. Também é provável o aparecimento de contatos em Estado 1, 2 ou 3 após estarem em Estado infinito.

tempo. Essa distância representa o número mínimo de saltos k necessário para que um dado nó estabeleça contato com outro.

As definições apresentadas anteriormente permitem analisar a vizinhança dos nós em uma rede móvel, estabelecendo seu comportamento e caracterizando o aproveitamento da k-vizinhança. As mudanças de estado nas vizinhanças dos nós, armazenadas nas linhas de tempo permitem estabelecer as probabilidades de transição entre estados, detalhando como os nós se movimentam uns em relação aos outros no tempo.

3. Definição do Problema

A busca pelo melhor aproveitamento da k-vizinhança em redes veiculares leva à natural investigação dos padrões de mobilidade, seja de um único nó isolado ou de um grupo de nós ao mesmo tempo. A partir dessa investigação, procura-se obter as melhores oportunidades de comunicação usando o maior número de contatos possível. Se os nós da rede apresentarem elevada mobilidade e o alcance das comunicações for limitado a distâncias curtas, é essencial levar em conta a velocidade relativa entre os nós. Dessa forma, evita-se considerar como oportunidades de comunicação os encontros com vizinhos cujo estado mais comum seja composto por muitos saltos, ou no pior dos casos, esteja em Estado infinito. Por exemplo, ao realizar a análise da vizinhança de todos os nós em uma topologia altamente dinâmica, é possível estabelecer as probabilidades de transição entre estados. Essas probabilidades são calculadas a partir das linhas de tempo, nas quais registram-se as mudanças entre estados para cada par de nós. A quantidade de transições de um Estado X para um Estado Y é contabilizada separadamente e posteriormente cada uma dessas estimativas é comparada com o total de transições ocorridas na rede, resultando nas probabilidades da Figura 2(a). Nessa figura, observa-se que as probabilidades mais significativas para o estabelecimento de contatos são encontradas a poucos saltos de distância. Além disso, a transição entre estados consecutivos é comum, mas diminui conforme o aumento da distância. Assim, é razoável inferir que a k-vizinhança é bem aproveitada utilizando-se apenas alguns saltos. Essa topologia dinâmica corresponde a um dos cenários analisados neste trabalho, chamado de Sintético. A análise completa se encontra na Seção 6.

Analisando novamente mesmo cenário Sintético, mas considerando agora a velocidade dos nós no momento do contato, alguns comportamentos mudam completamente. A Figura 2(b) mostra que se a velocidade relativa dos nós for menor do que 20 km/h, obtém-se um comportamento muito similar ao anterior. Com exceção da porcentagem de contatos que aparecem a um salto de distância, não existem diferenças significativas nas transições entre estados. Contudo, na Figura 2(c), observa-se que os resultados mudam radicalmente quando a velocidade relativa dos nós é superior aos 20 km/h. Essa figura mostra que a probabilidade de aparecimento de contatos é significativa somente para os primeiros quatro saltos, ou seja, nas transições do Estado infinito para os finitos, sendo desprezível para os demais casos (próxima de zero e por isso não representadas). Nota-se que essa probabilidade na transição do Estado infinito para o Estado 1 é muito superior a todas as outras.

A partir do comportamento verificado, este trabalho questiona se a k-vizinhança fornece na realidade alguma alternativa de comunicação quando existe mobilidade a velocidades mais altas, como no caso das redes ad hoc veiculares. Para tal, inclui-se a variável

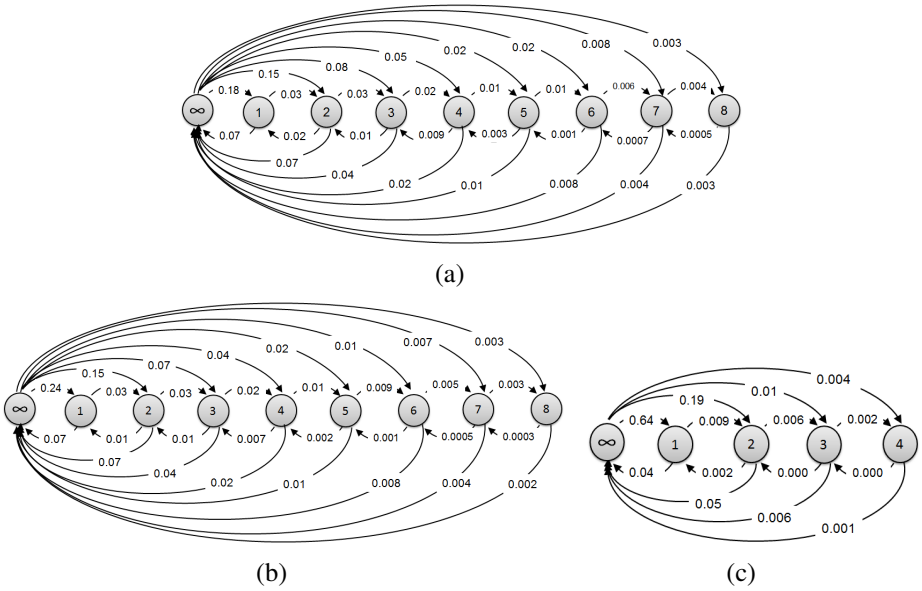


Figura 2. Probabilidades de transição entre estados do cenário Sintético (a) desconsiderando a velocidade dos nós, (b) para nós com velocidades menores do que 20 km/h e (c) para nós com velocidades maiores do que 20 km/h.

velocidade nas análises das comunicações a múltiplos saltos para que as chances reais de se usar essa estratégia de comunicação em ambientes comuns, como os ambientes urbanos convencionais, sejam verificadas. A partir dessa avaliação, pode-se investigar se os protocolos de comunicação que sempre assumem a existência de múltiplos saltos em redes móveis ad hoc, como o OLSR (*Optimized Link State Routing*) [Clausen e Jacquet, 2003], têm possibilidades reais de usar esses caminhos em cenários com velocidades relativas mais altas. Caso não tenham, esses protocolos podem apenas provocar uma sobrecarga desnecessária de pacotes de controle durante procedimentos como os de inundação.

4. Análise da Vizinhança Proposta

Este trabalho analisa a vizinhança levando em consideração a velocidade relativa com a qual os nós se movem em redes veiculares, para avaliar o aproveitamento da k-vizinhança, quando $k \geq 1$. A análise é realizada particularmente em cenários urbanos que geram topologias de rede bastante distintas e com diferentes densidades de nós.

A análise é feita em três cenários, cada um deles com características distintas de mobilidade, comportamento e forma de geração de traços. A metodologia usada para alcançar os objetivos dessa proposta consiste inicialmente na interpretação dos traços de mobilidade, seguida por uma análise de vizinhança a partir da obtenção das linhas de tempo entre todos os nós da rede. Considera-se que os nós entram em contato caso estejam dentro de um raio de cobertura fixo. São testados três valores distintos para esses raios, permitindo a comparação entre os resultados para raios diferentes. Em seguida avalia-se o eventual aproveitamento da k-vizinhança em função das diferentes velocidades relativas entre os nós. Por fim, uma simulação é realizada no simulador de redes NS-3 com um subconjunto de dados de um dos cenários usados na análise. O objetivo é demonstrar o impacto tanto na vazão da rede quanto na sobrecarga de controle, da premissa utilizada por diferentes protocolos de redes móveis referente à existência de caminhos de múltiplos

saltos, independente da velocidade dos nós móveis.

5. Conjuntos de Dados

Neste trabalho foram utilizados três traços com características distintas para diversificar a análise. Essas caraterísticas são apresentadas a seguir.

Mobility Dataset. Traços de 536 táxis recolhidos ao longo de 30 dias na cidade de São Francisco, Califórnia – EUA. O intervalo de tempo médio das atualizações é de aproximadamente 10 segundos [Piorkowski et al., 2009]. Esse traço representa um cenário de movimentação real, no qual as rotas seguidas pelos veículos não se encontram previamente estabelecidas e os horários para transitar pela cidade não são pré-definidos. Os dados fornecidos nesse traço foram analisados em sua totalidade.

TAPASCologne Project. Traço gerado pelo Instituto de Sistemas de Transporte do Centro Aeroespacial Alemão (ITS-DLR) com o objetivo de reproduzir a mobilidade veicular na área urbana da cidade de Colônia, na Alemanha. O traço sintético resultante reproduz de forma realista a mobilidade da cidade de Colônia, descrevendo as viagens de mais de 700.000 veículos individuais com granularidade de 1 segundo [Uppoor e Fiore, 2011]. O cenário representado por esse traço possui elevada densidade. Este trabalho analisa 10 minutos do total de 2 horas de registros.

Ad Hoc City Dataset. Contém registros da mobilidade de 1200 ônibus do sistema de ônibus de metrô da cidade de Seattle, Washington – EUA. Cada ônibus atualiza sua localização 536 vezes por dia [Jetcheva et al., 2003]. Esse traço mostra a movimentação real de uma frota de ônibus com caminhos pré-definidos para percorrer a cidade em horários estabelecidos. Este trabalho analisa registros de 2 dias do total de 5 dias disponibilizados.

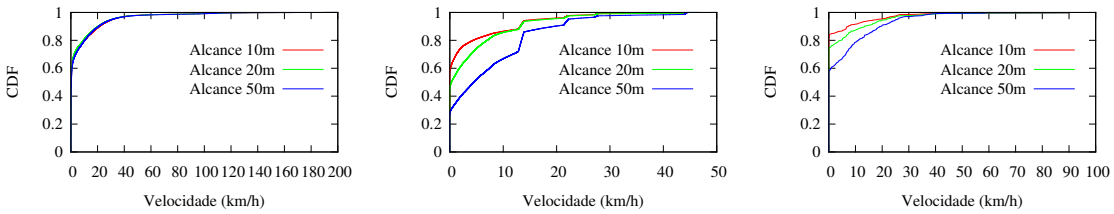
6. Análise da Vizinhança

A análise da vizinhança utiliza três distâncias diferentes para definir o raio de alcance máximo entre pares de nós. As distâncias de 10, 20 e 50 metros são utilizadas para avaliar o impacto do raio nos resultados. Vale mencionar que o raio de alcance é considerado fixo, pois essa informação não consta nos conjuntos de dados utilizados. Entretanto, demonstra-se que a variação do raio não afeta as conclusões obtidas. Além disso, como cada traço (Seção 5) constitui um cenário distinto para a análise, os resultados são identificados segundo o cenário e o raio de alcance utilizados. Assim, o primeiro conjunto de resultados engloba o cenário nomeado como “Táxis”, o segundo utiliza o cenário “Sintético”, e o último conjunto adota o cenário “Ônibus”. Os cenários foram nomeados segundo o tipo de veículo que compõe sua topologia e são referentes aos conjuntos de dados Mobility Data Set, TAPASCologne Project e Ad Hoc City Dataset, respectivamente.

A primeira etapa da análise caracteriza as distribuições de velocidade para cada um dos cenários estudados. Observa-se na Figura 3(a) que o cenário Táxis não apresenta mudanças significativas quando se varia o raio de alcance máximo. A quantidade máxima de veículos envolvidos nos contatos ocorre para baixas velocidades relativas, sendo que aproximadamente 90% dos contatos ocorrem para velocidades relativas inferiores a 20 km/h. Os 10% restantes dos veículos em contato se movem a velocidades relativas superiores a 20 km/h. Esses resultados são válidos para as três distâncias estudadas. Ressalta-se

que o limite superior das velocidades relativas dos veículos registradas no momento do contato não supera os 200 km/h. Já no cenário Sintético, visto na Figura 3(b), as velocidades relativas máximas no instante do contato não são superiores a 50 km/h, o que pode ser entendido como uma consequência das limitações dos cenários urbanos. Nota-se que o aumento da distância do raio de alcance máximo entre veículos provoca o aparecimento de contatos a velocidades relativas maiores. Esse fato também é consequência da maior densidade de carros no cenário, que favorece um maior aparecimento de contatos de um único e de múltiplos saltos. O cenário Ônibus é semelhante ao Sintético, isto é, o aumento da distância implica em maior número de contatos a velocidades relativas maiores, como visto na Figura 3(c). Novamente, observa-se que a maior porcentagem dos veículos envolvidos nos encontros se movem a baixas velocidades relativas para as três distâncias estudadas. Nesse cenário, as velocidades relativas máximas em que ocorrem contatos não superam os 100 km/h.

O comportamento das velocidades relativas nos três cenários para diferentes distâncias permite conhecer quais são as velocidades relativas entre um par de veículos no momento de contato. A avaliação desse comportamento permite concluir que, em três cenários característicos atuais, 90% dos contatos entre os nós acontecem a velocidades menores do que 20 km/h, e apenas 10% ocorrem em velocidades superiores.



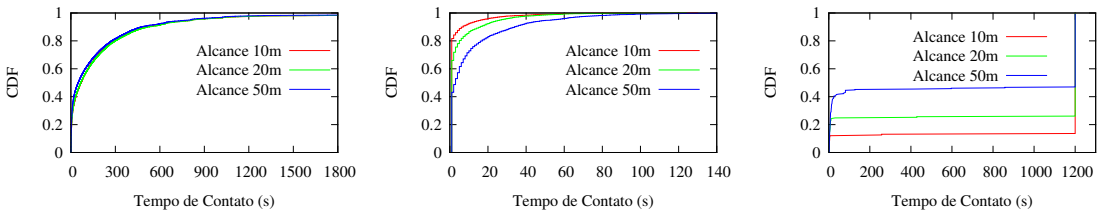
(a) Cenário Táxis. (b) Cenário Sintético (c) Cenário Ônibus

Figura 3. Distribuição das velocidades relativas.

A etapa seguinte da análise consiste em caracterizar os tempos de contato entre nós para os três raios de alcance definidos. O comportamento do cenário Táxis novamente é muito semelhante para os três raios de alcance, conforme visto na Figura 4(a). Nessa figura, nota-se que a maioria dos tempos de contato não supera os 600 segundos. No cenário Táxis existem muitos tempos de contato longos entre veículos com pouca ou nenhuma mobilidade. Acredita-se que esses resultados refletem bem o comportamento dos táxis que permanecem muito tempo parados em zonas específicas da cidade, enquanto aguardam passageiros. Os resultados na Figura 4(a) mostram uma pequena parcela de tempos de contato superior a 1800 segundos, referentes aos registros de veículos com mobilidade quase nula. O cenário Sintético fornece informação detalhada sobre as localizações e velocidades dos veículos com uma granularidade de 1 segundo. Dessa forma, é possível obter os tempos de contato entre os nós com maior exatidão, sendo o tempo mínimo registrado para cada contato igual a 1 segundo. A Figura 4(b) apresenta o comportamento desse cenário. Nessa figura, verifica-se que o aumento do raio acarreta em uma maior porcentagem de tempos de contato com maior duração, devido aos veículos permanecerem mais tempo dentro da zona de alcance mútuo. O cenário Ônibus exibe um comportamento singular para os resultados obtidos. Existem poucos contatos e a maioria deles ocorre entre veículos que estão parados. A Figura 4(c) mostra que esses contatos entre veículos

parados dura aproximadamente 1200 segundos. Ao variar o raio, a ocorrência de contatos também aumenta, mas o número de encontros entre veículos permanece baixo. Acredita-se que esses resultados representam de forma confiável o comportamento do tráfego de ônibus pelas ruas, uma vez que esses veículos possuem horários e rotas pré-estabelecidos, dificultando o estabelecimento de contatos entre eles durante a movimentação. Por outro lado, há também aumento dos tempos de contato quando estão parados na estação final.

A partir da análise realizada conclui-se que a frequência de ocorrência e a duração dos contatos variam muito em função da topologia representada pelo cenário. Essa variação acontece devido às diferenças na dinâmica de comportamento dos veículos e da densidade de veículos nos cenários.



(a) Cenário Táxis. (b) Cenário Sintético. (c) Cenário Ônibus.

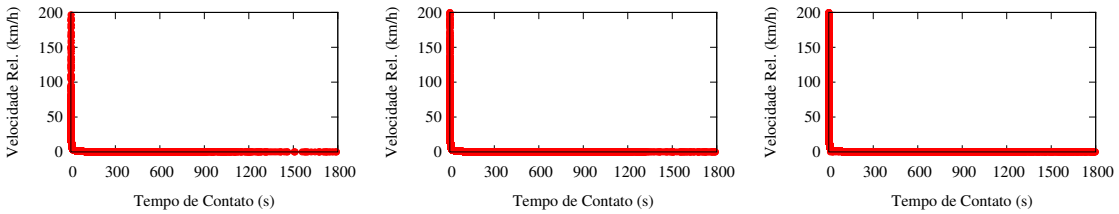
Figura 4. Distribuição dos tempos de contato.

A análise seguinte foi realizada com a finalidade de investigar o comportamento da velocidade relativa dos veículos em função do tempo de contato entre eles. No cenário Táxis, ilustrado na Figura 5, observa-se para os três raios de alcance, que quanto maior a velocidade relativa entre os nós, mais curtos são os contatos. Logo, além da probabilidade de estabelecimento de contatos a altas velocidades ser pequena, uma vez estabelecidos, eles apresentam curta duração. Velocidades mais baixas acarretam contatos mais longos, possibilitando um melhor aproveitamento da vizinhança.

O cenário Sintético, visto na Figura 6, possui maior número de pontos em que os contatos com duração significativa ocorrem a velocidades relativas elevadas, tornando-se mais evidente conforme o alcance aumenta. Contudo, mesmo existindo contatos duradouros a altas velocidades, a tendência observada é uma maior duração dos tempos de contato para velocidades relativas baixas, enquanto para altas velocidades essa duração diminui. O cenário Ônibus, visto na Figura 7, apresenta um comportamento similar. Apesar de existirem poucos encontros entre os ônibus, sobretudo em alcances menores, é notório que o tempo de contato entre os nós tende a aumentar para velocidades menores.

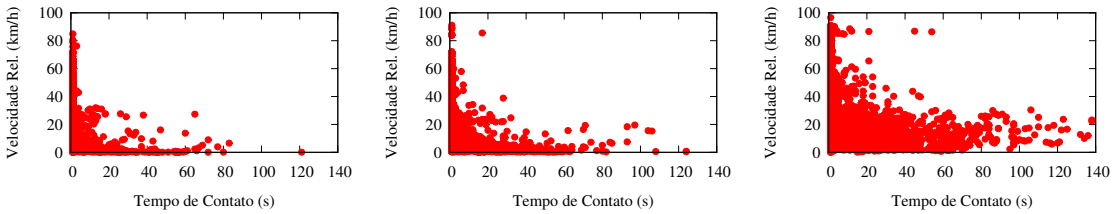
A análise da velocidade relativa em função do tempo de contato demonstra que as melhores oportunidades para aproveitamento da k-vizinhança concentram-se a baixas velocidades, quando os contatos são mais longos. A pouca movimentação em velocidades baixas caracteriza tipicamente situações urbanas de tráfego intenso. Veículos que se movimentam a altas velocidades tendem a apresentar contatos com duração muito curta, contribuindo pouco para o aproveitamento da vizinhança em comunicações em redes ad hoc veiculares. Essa característica evidencia que as oportunidades de contato a velocidades mais altas devem ser rapidamente aproveitadas, devendo-se evitar o desperdício de tempo com dados de controle para encaminhamento da informação. Essa constatação é por muitas vezes desconsiderada ou não tratada em trabalhos de

redes ad hoc móveis [Passarella e Conti, 2011]. Muitos protocolos de comunicação, como o OLSR [Clausen e Jacquet, 2003], ignoram esse fato e propõem estratégias para comunicações a partir de múltiplos saltos independente da velocidade dos nós e das condições típicas de operação da rede.



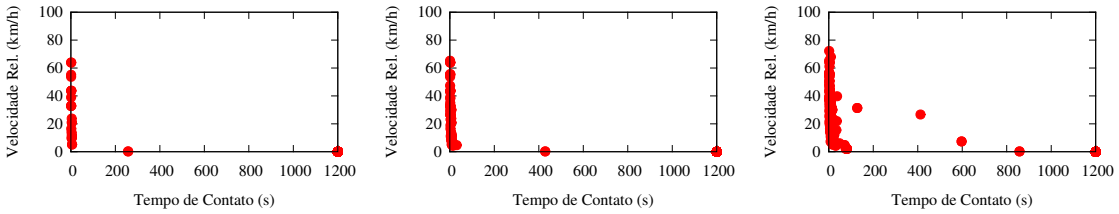
(a) Táxis, alcance = 10m. (b) Táxis, alcance = 20m. (c) Táxis, alcance = 50m.

Figura 5. Velocidade relativa pelo tempo de contato para o cenário Táxis.



(a) Sintético, alcance = 10m. (b) Sintético, alcance = 20m. (c) Sintético, alcance = 50m.

Figura 6. Velocidade relativa pelo tempo de contato para o cenário Sintético.



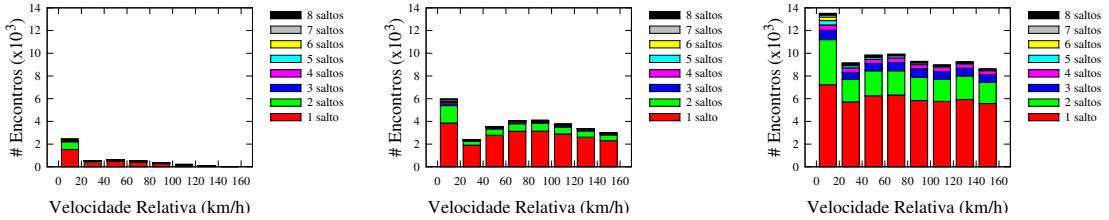
(a) Ônibus, alcance = 10m. (b) Ônibus, alcance = 20m. (c) Ônibus, alcance = 50m.

Figura 7. Velocidade relativa pelo tempo de contato para o cenário Ônibus.

Por fim, caracteriza-se o comportamento da k-vizinhança em função da velocidade relativa dos nós. A análise é realizada para uma distância de até oito saltos em diferentes faixas de velocidade para cada um dos cenários estudados. No cenário Táxis, conforme ilustrado na Figura 8, para um alcance de 10 metros, os nós tendem a estabelecer um número maior de contatos a velocidades relativas mais baixas. Além disso, o maior número de contatos ocorre a um salto de distância. Existe um crescimento significativo do número de contatos em velocidades mais altas, para os alcances de 20 e 50 metros. Contudo, mesmo com esse crescimento, o número de encontros a um salto de distância prevalece sobre qualquer outro estado. No cenário Sintético, como ilustra a Figura 9, o comportamento se altera. O maior número de contatos sempre é registrado para velocidades menores do que 20 km/h e a frequência dos estados com maior número de saltos se torna mais significativo. A presença expressiva de estados com maior número

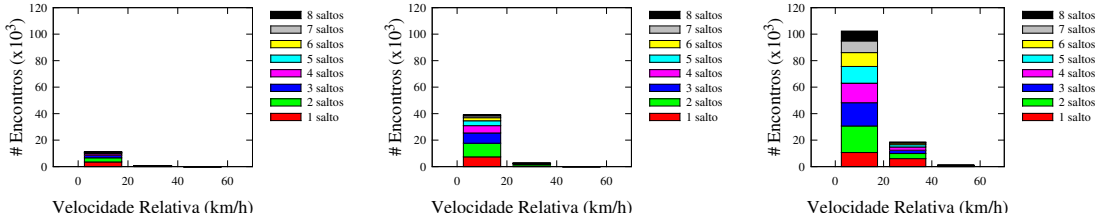
de saltos induz a acreditar que, no cenário estudado, pode-se explorar os múltiplos saltos da k-vizinhança para a comunicação. Porém, ao aumentar a velocidade, o número de encontros a múltiplos saltos diminui consideravelmente e, apesar de também ser reduzido, o número de contatos a um salto torna-se mais significativo. O comportamento predominantemente de existência de contatos a apenas um salto acontece também no cenário Sintético, apesar da maior densidade de nós. Esse argumento é importante já que a tendência do comportamento se repete em termos relativos independente da densidade de veículos no cenário. No cenário Ônibus ilustrado na Figura 10, a diferença no alcance da vizinhança é mais significativa. A análise para uma distância máxima de oito saltos retorna registros de contatos de até quatro saltos de distância, devido aos poucos encontros existentes entre os ônibus. De forma semelhante ao cenário Táxis, a maioria dos encontros registrados ocorrem a um salto. Observa-se ainda que conforme a velocidade relativa dos nós aumenta, o número de contatos diminui substancialmente.

Os resultados obtidos são reforçados pela análise dos tempos médios de permanência em cada estado. Essa análise indica que os contatos com maior duração tendem a ocorrer para os pares de nós com Estado 1, como visto na Tabela 1.



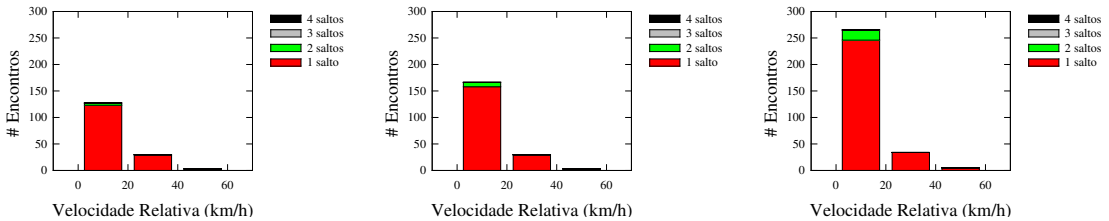
(a) Táxis, alcance = 10m. (b) Táxis, alcance = 20m. (c) Táxis, alcance = 50m.

Figura 8. k-vizinhança em função da velocidade relativa para o cenário Táxis.



(a) Sintético, alcance = 10m. (b) Sintético, alcance = 20m. (c) Sintético, alcance = 50m.

Figura 9. k-vizinhança em função da velocidade relativa para o cenário Sintético.



(a) Ônibus, alcance = 10m. (b) Ônibus, alcance = 20m. (c) Ônibus, alcance = 50m.

Figura 10. k-vizinhança em função da velocidade relativa para o cenário Ônibus.

Tabela 1. Tempo médio de permanência em cada estado em segundos.

Traços	Dist. (m)	Estado								
		∞	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Taxis	10	126924	165	44	11	5	2	1	1	1
	20	265117	926	385	231	179	145	120	103	86
	50	435689	2635	859	521	468	326	265	195	150
Sintético	10	13,8	5,5	2,6	2,1	1,7	1,3	1,1	0,9	1
	20	7,8	6	1,7	1,1	0,7	0,6	0,4	0,3	0,3
	50	5,9	9,5	2,1	1,2	0,8	0,6	0,5	0,4	0,3
Ônibus	10	5131	965	518	0,4	0,1	-	-	-	-
	20	4925	779	521	0,3	0,1	-	-	-	-
	50	4506	541	533	1,45	0,2	-	-	-	-

7. Estudo de Caso: Colônia

A fim de confirmar o estudo realizado sobre o aproveitamento da k-vizinhança em um cenário simulado, os procedimentos de inundação do OLSR foram avaliados. O OLSR foi escolhido justamente por utilizar procedimentos de inundação, mas qualquer outro protocolo com esta característica poderia ser usado. A ideia é verificar se existe realmente necessidade de inundar periodicamente redes em que os nós são móveis e entram em contato, na maioria dos casos, a distâncias de um salto. Com o objetivo de forçar os nós a evitarem a utilização de caminhos de múltiplos saltos, o TTL das mensagens de controle de topologia do OLSR foi configurado em 1. Tal modificação cancela o procedimento de inundação do OLSR, de forma que apenas os vizinhos de um dado nó recebem as mensagens de controle de topologia. Nesse caso, considera-se que mensagens de controle dificilmente irão alcançar vizinhos mais distantes e, se conseguirem, os caminhos formados em pouco tempo deixarão de ser válidos. Assim, assumir que as mensagens de controle podem alcançar todos os nós (TTL = 255) pode provocar sobrecarga na rede, sem trazer benefícios para a transmissão dos dados. Nas simulações, o traço sintético, disponibilizado pelo projeto TAPASCologne, foi utilizado no simulador de redes NS-3, versão 3.20. Atualmente, o NS-3 implementa módulos relacionados ao padrão IEEE 802.11p, desenvolvido especialmente para comunicação entre veículos, possibilitando uma simulação mais realista para o cenário das redes veiculares. Além disso, esse simulador permite a escolha de parâmetros de modelagem do meio sem-fio, como atenuação e atraso de propagação do canal, potência de transmissão, limiar de recepção e frequência de operação dos dispositivos móveis. Por esses motivos o NS-3 foi utilizado.

O traço de Colônia completo apresenta 2h de informação, tendo sido dividido em 12 partes de 10 minutos cada. Na análise da Seção 6, os primeiros 10 minutos foram utilizados. A simulação usa uma amostra representativa desse subconjunto, para a qual o comportamento das métricas avaliadas permanece aproximadamente constante. Escolheu-se uma pequena área de aproximadamente 4,8 km^2, contendo 688 nós, que se movem pela região durante 10 minutos. O protocolo OLSR foi instalado em cada um dos nós.

7.1. Parâmetros de Simulação

Os parâmetros de configuração da camada física e os atributos dos modelos de propagação foram escolhidos com base nos trabalhos de [Cheng et al., 2007, Singh, 2012, Yin et al., 2006, Islam et al., 2013, Mangel et al., 2011]. Esses parâmetros foram modificados para se adequarem à realidade do cenário urbano, conforme valores típicos do padrão IEEE 802.11p. Esse padrão limita a EIRP (*Effective Isotropic Radiated*

Power) dos dispositivos a 23, 33 ou 44,8 dBm, dependendo da sua classificação. Neste trabalho, a potência de transmissão dos dispositivos foi configurada para 23 dBm, com ganho de antena nulo e sem inclusão de perdas no equipamento transmissor, de forma que a EIRP de cada nó é igual a 23 dBm, simulando o pior caso de limitação superior. A sensibilidade do receptor foi configurada para -95 dBm, tendo sido definida com base na sensibilidade de produtos disponíveis no mercado e em conformidade com os padrões internacionais do IEEE para redes veiculares [Arada Systems, 2013]. A frequência de operação foi configurada para 5,9 GHz, estando dentro da faixa de frequências padronizada para uso em redes veiculares.

O canal de comunicação foi modelado para se adequar ao ambiente urbano. Para tanto, foi utilizado o modelo de propagação de Nakagami-m, que permite modelar o desvanecimento rápido em um canal sem-fio, em conjunto com o 3-Log-Distância, que permite que sejam definidos expoentes de atenuação distintos por faixa de distância entre transmissor e receptor. A Figura 11 define os valores dos parâmetros de forma e de atenuação dos modelos de Nakagami-m e 3-Log-Distância, respectivamente. Além disso, a Figura 11 também determina as distâncias em que esses parâmetros são válidos. Cada região é caracterizada de acordo com a intensidade dos efeitos provocados pela atenuação e pelo desvanecimento. A zona de espaço livre caracteriza-se por apresentar perdas no canal devido principalmente à atenuação [Cheng et al., 2007] ("conjunto de dados 1"). Na zona de Rician o sinal passa a sofrer maior influência do desvanecimento e maior atenuação. Os efeitos desses mecanismos de propagação são intensificados na região de Rayleigh, e a zona seguinte caracteriza-se por apresentar grande degradação da relação sinal-ruído. No modelo de propagação 3-Log-Distância a perda de referência foi configurada em 61,84 dB, considerando a frequência de operação de 5,9 GHz e o alcance de 5 metros para comunicação em espaço livre.

A camada de enlace foi configurada com a instalação padrão, na qual a banda de transmissão é igual a 10 MHz, com controle de taxa constante e modulação OFDM com taxa de 6 Mb/s. A comunicação entre os nós foi realizada através da instalação de aplicações UDP cliente/servidor. Nessa instalação escolheu-se aleatoriamente, dentre os 688 nós, N nós para serem clientes e outros N nós para se rem servidores, com N variando de 20 a 300. A instalação nos nós foi feita de forma que o cliente C_i se comunique apenas com o servidor S_i, sendo essas instâncias residentes em nós distintos.

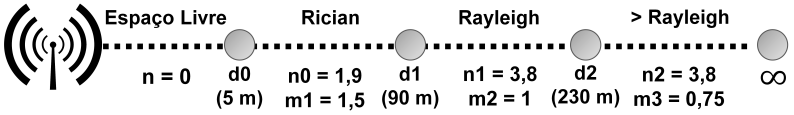


Figura 11. Valor dos parâmetros dos modelos de propagação Nakagami-m e 3-Log-Distância e representação das regiões influenciadas pela atenuação e pelo desvanecimento.

7.2. Resultados da Simulação

Os pontos apresentados na Figura 12 foram calculados a partir da média dos resultados obtidos através das simulações, considerando-se um intervalo de confiança de 95%. A Figura 12(a) mostra que, ao variar o número de aplicações instaladas no cenário, apesar do processo de inundação levar teoricamente ao estabelecimento de caminhos mais

próximos do ótimo, a vazão da rede não é beneficiada. Essa observação advém do fato de as curvas para ambos os valores de TTL serem estatisticamente iguais, comprovando que para o cenário estudado os procedimentos de inundação apenas acrescentam sobrecarga de controle à rede, sem trazer benefícios para a vazão da rede. Esse efeito é consequência da duração dos tempos de vida dos caminhos formados através da inundação de mensagens de controle de topologia ser menor em cenários altamente móveis, caracterizados por apresentarem mudanças de topologia muito rápidas e frequentes. Cada modificação na topologia pode invalidar inúmeras rotas que utilizam múltiplos saltos, de forma que os procedimentos de inundação perdem sua eficiência, a qual consiste em permitir o conhecimento das possíveis rotas de antemão.

O resultado obtido está em acordo com o observado na análise da Seção 6, no qual a maioria dos contatos são estabelecidos a apenas um salto, de forma que os dados de controle utilizados pelos procedimentos de inundação de alguns protocolos de redes móveis ad hoc provocam uma sobrecarga desnecessária na rede. A Figura 12(b) ilustra quão intensa essa sobrecarga pode ser. Por exemplo, para grande parte dos pontos analisados, a quantidade de pacotes de controle obtidos para TTL 1 cresce de uma ordem de grandeza de 10^6 quando medidos para TTL 255 para um mesmo número de aplicações. Portanto, a eficiência das transferências de dados deve ser melhorada, sendo importante aproveitar os contatos em todas as ocasiões favoráveis, de forma oportunística.

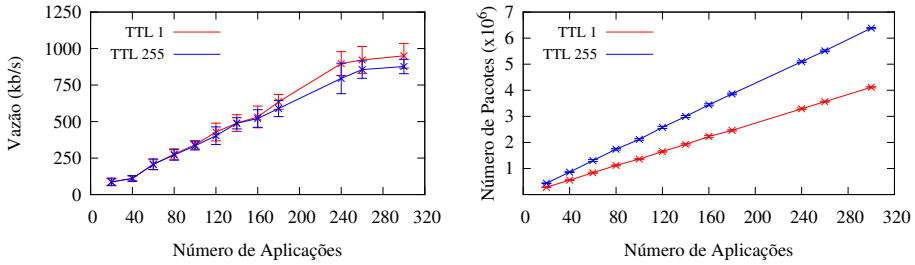


Figura 12. Resultados da simulação para o traço de Colônia variando-se o número de aplicações instaladas no cenário. (a) A vazão total da rede não se beneficia pelo uso de TTL igual a 255, que apenas serve para gerar (b) sobrecarga de roteamento no cenário investigado.

8. Conclusões

A análise deste trabalho permite concluir que o aumento da velocidade dos nós influi negativamente no estabelecimento de caminhos de comunicação em redes móveis ad hoc, particularmente em redes veiculares. Assim, para as redes estudadas neste trabalho, o uso de caminhos de múltiplos saltos não representa uma alternativa eficiente quando existem velocidades relativas mais elevadas. Nesses casos, todo esforço em busca de caminhos de múltiplos saltos pode apenas representar um desperdício de banda passante. A análise de traços reais e sintéticos permitiu verificar que os contatos ocorrem predominantemente a apenas um salto de distância e, através de simulações, foi verificado que inundar redes móveis pode apenas representar uma sobrecarga adicional de controle. Como trabalhos futuros, pretende-se estender a análise realizada para outros cenários, investigar com maior profundidade os efeitos da densidade de nós no comportamento da vizinhança e avaliar outros raios de alcance e protocolos de roteamento para redes veiculares.

Referências

Arada Systems (2013). LocoMate Products. Acessado 12/2014 em http://www.aradasystems.com/products/.

Cheng, L., Henty, B., Stancil, D., Bai, F. e 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 JSAC*, 25(8):1501–1516.

Clausen, T. e Jacquet, P. (2003). Optimized link state routing protocol (OLSR). RFC 3626.

Conan, V., Leguay, J. e Friedman, T. (2007). Characterizing pairwise inter-contact patterns in delay tolerant networks. Autonomics '07, p. 19:1–19:9.

Gonzalez, M. C., Hidalgo, C. A. e Barabasi, A.-L. (2008). Understanding individual human mobility patterns. *Nature*, 453(7196):779–782.

Islam, T., Hu, Y., Onur, E., Boltjes, B. e de Jongh, J. (2013). Realistic simulation of IEEE 802.11p channel in mobile vehicle to vehicle communication. COMITE '13, p. 156–161.

Jetcheva, J. G., Hu, Y.-C., PalChaudhuri, S., Saha, A. K. e Johnson, D. B. (2003). CRAWDAD data set rice/ad_hoc_city (v. 2003-09-11). Acessado 09/2014 em http://crawdad.org/rice/ad_hoc_city/.

Mangel, T., Klemp, O. e Hartenstein, H. (2011). A validated 5.9 GHz non-line-of-sight path-loss and fading model for inter-vehicle communication. ITST '11, p. 75–80.

Olariu, S., Yan, G. e Salleh, S. (2010). A probabilistic routing protocol in VANET. *IJMCMC*, 2(4):21–37.

Passarella, A. e Conti, M. (2011). Characterising aggregate inter-contact times in heterogeneous opportunistic networks. NETWORKING'11, p. 301–313.

Phe-Neau, T., Campista, M. E. M., Dias De Amorim, M. e Conan, V. (2013). Padrões de Mobilidade de Vizinhança em Redes de Contato Intermitente. SBRC '13, p. 397–410.

Phe-Neau, T., Dias de Amorim, M. e Conan, V. (2012). Vicinity-based DTN characterization. MobiOpp '12, p. 37–44.

Piorkowski, M., Sarafijanovic-Djukic, N. e Grossglauser, M. (2009). CRAWDAD data set epfl/mobility (v. 2009-02-24). Acessado 05/2014 em http://crawdad.org/epfl/mobility/.

Rezende, C. G., Pazzi, R. W. e Boukerche, A. (2009). An efficient neighborhood prediction protocol to estimate link availability in VANETs. MobiWAC '09, p. 83–90.

Singh, P. K. (2012). Influences of TwoRayGround and Nakagami propagation model for the performance of adhoc routing protocol in VANET. *IJCA*, 45(22):1–6.

Uppoor, S. e Fiore, M. (2011). Large-scale urban vehicular mobility for networking research. VNC '11, p. 62–69.

Yin, J., Holland, G., ElBatt, T., Bai, F. e Krishnan, H. (2006). DSRC channel fading analysis from empirical measurement. ChinaCom '06, p. 1–5.

Zhao, Z., Mosler, B. e Braun, T. (2012). Performance evaluation of opportunistic routing protocols: A framework-based approach using OMNeT++. LANC '12, p. 28–35.

Booter websites characterization: *Towards a list of threats*

Justyna Joanna Chromik, José Jair Santanna, Anna Sperotto, and Aiko Pras

[1]University of Twente - The Nederlands
Design and Analysis of Communication Systems (DACS)

j.j.chromik@student.utwente.nl,{j.j.santanna,a.sperotto,a.pras}@utwente.nl

Abstract. *Distributed Denial of Service (DDoS) attacks mean millions in revenue losses to many industries, such e-commerce and online financial services. The amount of reported DDoS attacks has increased with 47% compared to 2013. One of the reasons for this increase is the availability and ease of accessibility to websites, which provide DDoS attacks as a paid service, called Booters. Although there are hundreds of Booters available, current researches are focused on a handful sample of them - either to analyse attack traffic or hacked databases. Towards a thorough understanding and mitigation of Booters, a comprehensive list of them is needed. In this paper we characterize Booter websites and demonstrate that the found main characteristics can be used to classify Booters with 85% of accuracy. The Dutch National Research and Education Network (SURFnet) has been using a list generated by our methodology since 2013, what demonstrates high relevance to the network management community and the security specialists.*

1. Introduction

Distributed Denial of Service (DDoS) attacks are considered number one operational threat on the Internet. DDoS attacks aim to make a target machine, service or network unavailable to its intended users. To perform powerful attacks, attackers (mis)use hundreds or even thousands of distributed sources, such as infected computers or misconfigured servers. For many industries, such as e-commerce and online financial services, DDoS attacks are especially devastating. To those industries, DDoS attacks cause millions in revenue losses, reputation damage, and customer attrition [Arbor Networks 2014, Ponemon Institute 2014].

The amount of reported DDoS attacks has increased with 47% compared to 2013 [Akamai Technologies 2014]. One of the reasons for this increase is the effectiveness, simplicity and availability of websites that provide DDoS attacks as a paid service, called *Booters*. These websites offer attacks for a very cheap price, for instance, less than 5 USD [Karami and McCoy 2013], powerful enough to put most of small and medium-sized enterprise companies' websites offline [Santanna et al. 2015b]. Researches on mitigation of Booters phenomenon focus basically in two areas: (i) characterising the attacks [Santanna et al. 2015b] and (ii) analysing the leaked databases [Santanna et al. 2015a]. Although both of them have a clear contribution towards Booter mitigation, in order to address the whole phenomenon (not a particular set of Booters), they fail in a key element: a relevant, updated, and extensive list of Booters.

In order to help security specialists to retrieve such list and get a thorough understanding about the Booter phenomenon, the goal of this paper is to reveal the main characteristics of Booter websites. By using those characteristics we show that a Booter list can be retrieved with high accuracy. The main contributions of this paper are the following:

- To provide a relevant set of search terms to retrieve URLs related to Booters and a crawler to retrieve all needed information of the websites, such as the structure, visual and textual content, and the WHOIS information (Section 3).
- To reveal the 9 main characteristics of Booter websites, achieved by an extensive analysis of features used to classify websites (Section 4).
- To demonstrate that the 9 main characteristics of Booter websites can classify Booter with an accuracy of 85% (Section 4.6).

Since 2013, a Booter list generated based on our methodology have been used by the Dutch National Research and Education Network (SURFnet) to observe the users accessing Booters and their correlation with attacks. This is just one example that demonstrates high relevance of our work to the network management community and the security specialists. Only by knowing the threats we can mitigate them.

2. Related work

In the literature there is no work related to Booter websites classification. Therefore, in this section we focus on approaches that classify generic websites based on features. Although the existing approaches have several different goals we focus on specifically three of them: (i) approaches that classify the subject or type of a website [Lindemann and Littig 2006, Rajalakshmi and Aravindan 2011, Lindemann and Littig 2007, Kovacevic et al. 2004], (ii) approaches that aim to filter websites [Jo et al. 2013, Hammami et al. 2006], and (iii) approaches that aim to generate blacklists [Ma et al. 2009]. The features used in those approaches could possibly be used in Booter area. The list of features that we found is the following:

1. **URL**: this feature discloses the overall composition of a URL. Usually a URL is composed of three elements: (i) network application protocol, (ii) the URL host name, and (iii) the URL path name. For example, in *http://www.domainname.tld/path/to/the/article.html*, 'http://' is the protocol, 'www.domainname.tdl' is the host name, and '/path/to/the/article.html' is the path name.
2. **Website structure**: is a feature that reveals, among others, two aspects: (i) the website depth level and (ii) the number of known pages of a website. The former is a terminology defined by us that analyses the number of slashes ('/') a URL path has. For example, the URL *http://www.domainname.tld/path/to/the/article.html* has depth level '4'. The latter aspect is the number of indexed webpages that have the same host name and are reached by Google Search engine. For example, the number of know pages is 2 if Google Search returns 2 pages that contain a same *www.domainname.tld*, such as *www.domainname.tld/1.html* and *www.domainname.tld/2.php*.
3. **WHOIS**: is a feature that reveals information of domain names, such as (i) the registration date, (ii) the owner, (iii) the nameservers related to that domain name, and (iv) the entity responsible for the domain registration (i.e., registrar).

4. **Page content**: is a feature that reveals the elements of a website such as (i) the textual description, (ii) the meta data, and (iii) the visual content, (e.g., buttons, tables, and figures).

In Table 1 we summarize the features that each work addresses to classify websites. Such table shows that most of the existing works focus on the URL, website structure, and the website content. WHOIS and visual content are less popular, but definitely worth investigating. Therefore, in the next section we investigate those four features to characterize Booters.

Paper	URL	Website structure	WHOIS	Website content		
				Textual	Meta	Visual
[Lindemann and Littig 2006]		x			x	
[Rajalakshmi and Aravindan 2011]	x					
[Lindemann and Littig 2007]	x	x				
[Jo et al. 2013]	x		x	x	x	
[Hammami et al. 2006]		x		x	x	x
[Ma et al. 2009]	x	x	x			
[Kovacevic et al. 2004]						x

Table 1. Website features used to characterize websites.

3. The search terms, legal considerations, and our crawler

To investigate the four features described in the previous section, firstly we describe our methodology to define a list of search terms that we use to retrieve Booter websites. We give a special attention to the first search term that we found: "stresser", because it lead us to some legal considerations. After that we describe a crawler that we develop to investigate Booter websites features.

3.1. The first search term and legal considerations

In order to find a list of search terms, with which we can retrieve Booter websites, we started using the most obvious term: "booter". By using such term we found many URLs related to the word "stresser". By definition, Booters should be different from stressers. Booter is a website that provides DDoS attacks as a paid service [Santanna and Sperotto 2014], while by definition a stresser is a company (usually accessed via a website) that provides stress testing on a given system [Rouse 2007].

Both definitions are almost the same because performing a DDoS attack is a way to "stress test" a target system. However, note that stressers aim to intentionally test their customers' systems beyond regular operational capacity, while Booters are used by their clients to "stress test" a third party service. Another important characteristic is that (in theory) stressers perform their tests deliberately and in a test lab, not affecting other entities.

Putting the theoretical definitions aside, we compare the services offered by both, a Booter and a stresser. To do so, we analyse the top one result for the search terms "booter" and "stresser". We decide to anonymize the URL/names of the Booter and the stresser to avoid legal or ethical implications. Table 2 shows our findings.

Attack types		Booter A	Stresser B
Layer 3&4	TCP	✓	
	SSYN	✓	✓
	ESSYN	✓	
	UDP	✓	✓
	UDP-LAG	✓	✓
	DRDoS		✓
	CHARGEN	✓	
Application layer	GET	✓	
	HEAD	✓	
	POST	✓	
	RUDY	✓	✓
	ARME	✓	✓
	SLOWLORIS	✓	✓

Table 2. Services offered by a Booter and a stresser

The first observation is that technically the Stresser B offers the same resource exhaustion actions (attacks) as Booter A. The second observation is that both offer most of the known types of DDoS attacks, which target the Layers 3&4 and the application layer. Our paper has no intention to describe how each type of attack works, for more information see [Mirkovic and Reiher 2004]. The third observation is that although Booter A does not explicitly offer Distributed Reflection Denial of Service (DRDoS), CHARGEN is a type of DRDoS. Therefore, both Booter A and Stresser B offer the strongest type of DDoS attacks reported nowadays. Finally, we notice that Stresser B has no restriction in relation to the target of "stress test". Given this freedom, the customers can perform attacks against third party services. Therefore, through these four observations we conclude that Stresser B is also a Booter.

The question that arise is: why Booters advertise themselves as stressers? The answer, which we found in hacker forums and many blogs, is that Booters want to avoid legal problems by hiding illegal actions (DDoS attacks using compromised machines) behind legal services (stress testing) [Kassner 2013, Musthaler 2012]. Booters have another strategy to avoid legal problems. Instead of advertising themselves as stressers, they include in their websites Terms of Services (ToS). It is a legal agreement composed by a set of rules that clients need to follow to use their services. Table 3 shows parts of ToS found on Booters.

	Terms of Service statements
Booter C	"We are not responsible for how ever you use this stresser"
Booter D	"Illegal activity which occurs in your account is [...] associated to you"
Booter E	"Anything you do while on Booter E is your own responsibility"

Table 3. Examples of text included in Terms of Service

We observe in Table 3 that Booters clearly do not take any legal responsibilities for user's actions, although their core business is to offer DDoS attacks against anyone and anything connected to the Internet. The legal aspect of Booters is a nudging subject, which requires attention, but will be out of the scope of this research. From technical point

of view we see that stressers are used as Booters. Because of this we use term "stresser" in the set of terms to retrieve Booter websites. In the next subsection we describe how did we find the other search terms.

3.2. The other search terms

To find other terms we search on Google for materials related to Booters and stressers. Through an extensive literature we found other two terms: *"ddos-for-hire"* [Krebs 2013a] and *"ddoser"* [Safe Keys 2013]. Finally, by using the two former terms ("booter" and "stresser") and the later terms ("ddos-for-hire" and "ddoser") we search the literature using Google Scholar. This Google service aggregates most of the the digital libraries, academic publishers, and repositories worldwide, including IEEE Xplore Digital Library and ACM Digital Library. Through Google Scholar, and closing our set of terms, we found "ddos-as-a-service" [Karami and McCoy 2013]. Note that the reference on the side of earlier mentioned terms is not necessarily the first to use/define these terms, but the place that we have found them.

3.3. Our crawler

After defining the five search terms (*"booter", "stresser", "ddos-for-hire", "ddoser", and "ddos-as-a-service"*), in this section we describe our approach to retrieve a list of URLs related to Booters and the additional information needed to classify them according to the four website features (described in Sec. 2). To do so, we develop a crawler to fulfil the following requirements:

1. Retrieve as many URLs related to Booters as possible based of the list of search terms. These retrieved URLs are called in this paper as "potential Booters".
2. Extend the URLs retrieved in the previous requirement to include all known pages related to these URL domain names.
3. Retrieve the WHOIS information of the URL domain names.
4. Download the Booter website content.

In order to retrieve the URLs related to Booters (first requirement), we use a python search script [1]. This approach was chosen over other approaches [2][3][4] because it is not limited on the amount of results able to retrieve (e.g., 24 URLs). To find all web pages related to a website (second requirement) we use the Google Search with operator "site:". This operator returns as result only URLs within the domain requested, for example the output of *"site:name-of-potential-booter.com"* is a list of pages (URL paths) in the domain *"name-of-potential-booter.com"*.

To retrieve the WHOIS information (third requirement) we use pythonwhois package [5]. Although such library is able to return more than 20 different information about a domain name, in our WHOIS analysis (Sec. 4.4) we only use 4 of them: (i) the creation date, (ii) the registrar of the domain name, (iii) the nameservers that the domain is pointing to, and (iv) the contacts of the domain administrator. To fulfil the last requirement and

[1] http://breakingcode.wordpress.com/2010/06/29/google-search-python/
[2] http://googolplex.sourceforge.net/
[3] http://www.catonmat.net/blog/python-library-for-google-search/
[4] http://googlesystem.blogspot.nl/2008/04/google-search-rest-api.html
[5] http://cryto.net/pythonwhois/

download the content of Booter websites, we firstly decide that we should keep our identity anonymous. This precaution was taken because a security specialist was successively attacked after starting investigating Booters [Krebs 2013b]. Therefore, we use The Onion Router (TOR) network that enables online anonymity by encrypting and bouncing the traffic through networks and open relays severs. Secondly, another requirement needed to download the potential-booter-webpages is to support JavaScript. To do so, we use the Selenium browser[6], which is a library that emulates a generic Web browser and therefore is able to support JavaScript.

In summary, our crawler was build to attend the requirements to analyse Booters. Although we use specific libraries and APIs to build our crawler, these were our decisions on how to fulfil the requirements. Therefore other libraries can be used. Note that our crawler can be extended to perform an automated classification of Booter websites, but first the characteristics of the website features should be analysed. More ideas about future works are written in the last section of this paper.

4. Analysis on Booter websites features

In this section we analyse the features of Booter websites. Firstly we describe the list of URLs (Sec. 4.1) used to perform our analyses. Secondly, we describe our analysis based on the URL characteristics (Sec. 4.2), the website structure (Sec. 4.3), the WHOIS information (Sec. 4.4), and the website content (Sec. 4.5). We close this section highlighting the features that are more representative to classify Booters (Sec. 4.6).

4.1. The list of URLs

By using the search terms and our crawler described in the previous section (Sec. 3.3) we retrieved 1238 URLs, on 15th September 2014. From these 1238, 230 URLs were retrieved by using the term "booter", 370 by using "stresser", 265 "ddoser", 199 "ddos-for-hire", and 174 "ddos-as-a-service". In order to verify if these search terms are representative and distinct we analyse the intersection of retrieved URLs, showed in Table 4.

Search Terms	booter	stresser	ddoser	ddos-for-hire	ddos-as-a-service
booter	**230**	13	3	1	1
stresser	13	**370**	3	0	0
ddoser	**3**	3	**230**	1	1
ddos-for-hire	1	0	1	**174**	11
ddos-as-a-service	1	0	1	11	**199**

Table 4. Intersection of retrieved URLs based on different search terms

Based on Table 4 we notice that our search terms are distinct: very small number of same URLs are retrieved using different search terms, for example only 3 URLs were retrieved using the terms "booter" and "ddoser". " We are aware that our crawler did not retrieve all URLs related to those search terms because each search process was interrupted by a HTTP error (i.e., 503: service unavailable). Although our chosen approach retrieves more URLs than other current approaches, Google Search is still able to detect and block our crawler. Even though, the retrieved list of URLs is sufficient enough to analyse Booters features, since it contains both Booter and non-Booter websites.

[6]http://www.seleniumhq.org

4.2. URL analysis

Based on the list with 1238 URLs we analyse the first feature found in our survey: the URL composition. Table 5 shows a summary of the types of URLs found.

URL Type	URL	# URL retrieved
1	potential-booter.com	71
2	potential-booter.com/login.php	1167
3	www.domain.com/potential-booter	
4	potential-booter.domain.com	

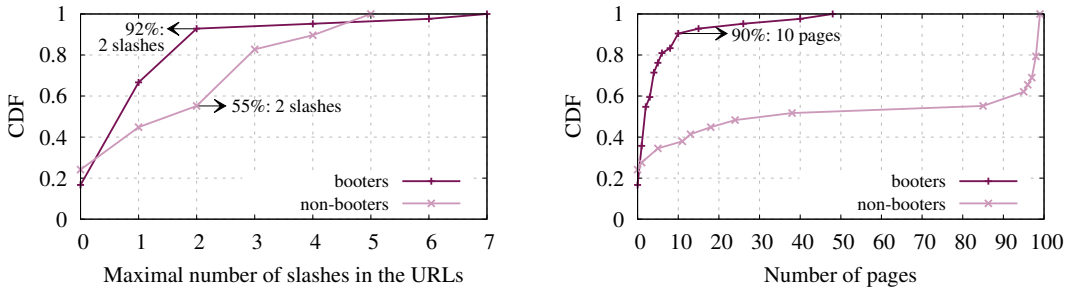
Table 5. Examples of retrieved URLs

In Table 5 we observe four different types of URLs: with only a host name (type 1), with host name and path (type 2), where Booter is a page of a website (type 3), and where a Booter is a subdomain of a domain name (type 4). We want to focus on the URL type that has the highest probability to be a Booter. Note that often URLs type 2 are subpages of URLs type 1. Thus, we decide to analyse only the type 1. Although URLs type 3 and 4 can potentially contain Booters, we noticed that it is more likely that they provided information about Booters, not the websites themselves.

Each one of the 71 URLs type 1 are later called as *potential Booter*, and they will be further classified. After a manual analysis of those potential Booters we found that 42 are Booter websites and 29 are non-Booter websites. All the further sections will be based on the 71 potential Booters.

4.3. Website structure analysis

After the filtering process described in the previous section, we analyse the structure of these 71 potential Booters. Figure 1 summarizes our findings.



(a) Depth of websites.

(b) Known pages.

Figure 1. CDF of website structure aspects.

In Figure 1, graphs of the Cumulative Distribution Function (CDF) of the website structure are shown. While Figure 1(a) shows the CDF of depth level of these 71 website, Figure 1(b) shows the CDF of the known pages. Both graphs show the results for Booters and non-Booters (classified manually in the previous section). We observe in Figure 1(a) that 92% of Booters have their website depth up to 2 levels, whereas only 55% of non-Booters have the same depth. Considering Figure 1(b) we observe that 90% of Booters have 10 or less pages. In addition, Booter websites never exceed 50 known pages, what is

an interesting observation that can be used to eliminate non-Booters from a set of URLs (that in this analysis is almost 50%).

By in depth analysis of the 71 potential Booters we notice that some page names appear more than once, such as “register”, “ToS”, “plans”, “buy”, and “hub”. The number of times we observed those pages is shown in Figure 2. We notice that pages as Terms of Service (ToS) and “registration” are more often appearing when analysing Booters. Almost 60% of Booters have a registration page (24/42) and around 40% of Booters have ToS page (17/42).

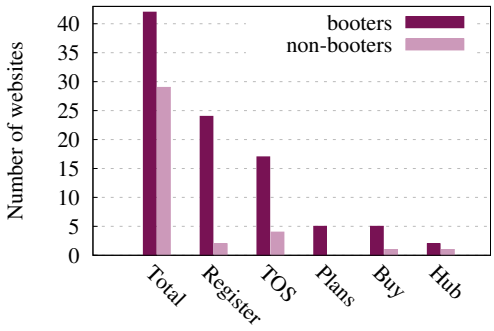


Figure 2. Page names analysis.

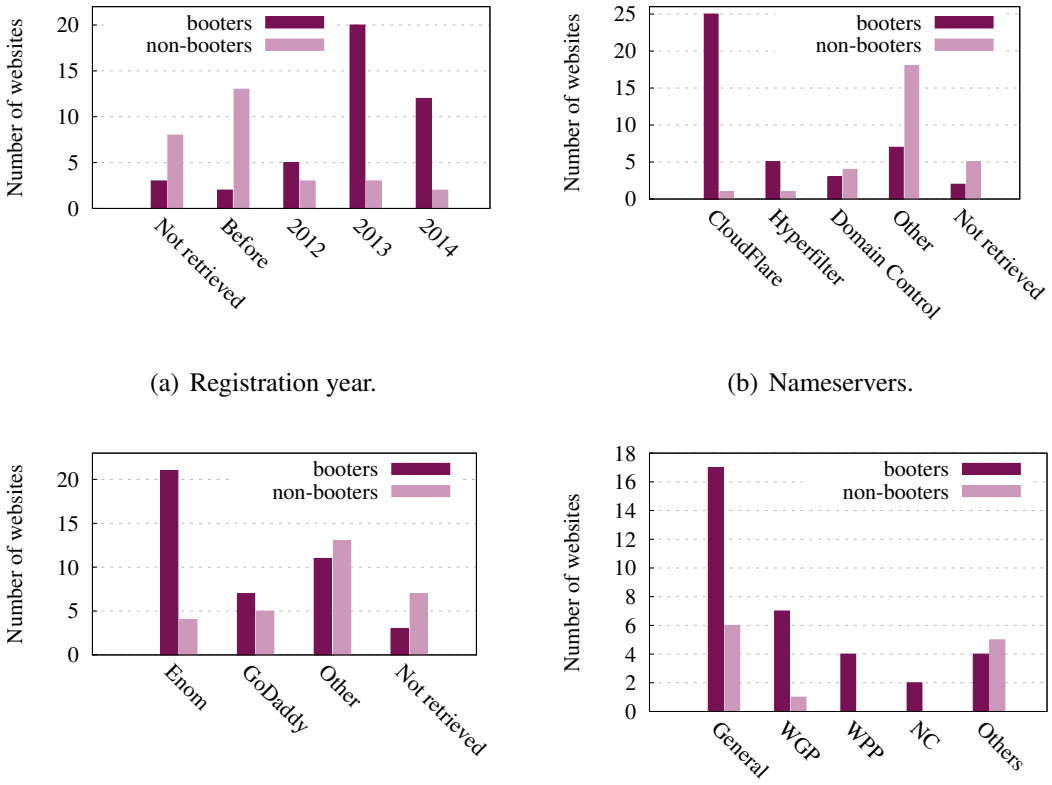
The other page names on the Figure 2, plans, buy, and hub, are not that promising. For example only 5 out of 42 Booters had pages referring to plans or redirecting to purchase page (buy). Although the page “hub” is crucial to Booters, because through that page a user can start an attack, it was found only twice. The reason for that could be that this page is accessible only after logging into the Booter and it is not indexed by Google.

4.4. WHOIS analysis

After analysing the website structure we check the WHOIS information of the 71 potential Booters retrieved by our crawler. We present our findings in Figure 3.

In Figure 3 we observe four different information related to WHOIS. From the first one, in Figure 3(a), we observed that more than 88% of the Booter domain names (37/42) were registered in 2012 (5), 2013 (20), and 2014 (12), whereas a bit less than half of the non-Booter websites were registered before 2012 (13/29). Overall, it shows that most of the Booters by our crawler are relatively new. By analysing the namesevers of the retrieved Booter domain names, showed in Figure 3(b), we are not surprised to observe that almost 60% of Booters (25/42) are associated to CloudFlare. It was already pointed out in [Santanna et al. 2015b] that Booter subscribe services from CloudFlare to be protected against DDoS attacks from the competitors (other Booters). However, we are surprised to observe another company offering DDoS protection, Hyperfilter. It is the first time that this company is observed in this phenomenon. Also in Figure 3(b), we confirm that majority of non-Booters have more variety in nameservers (18/29).

By analysing the companies that provide the domain registration (registrar), showed in Figure 3(c), we notice that half of Booters (21/42) have their domain registered with Enom, which is a known domain registrar. This is probably because of cheap



(a) Registration year.

(b) Nameservers.

(c) Registrar.

(d) Contact of the domain name owner.

Figure 3. WHOIS information.

price, but also because Enom offers the "WhoisGuard" service used for hiding the contact details. This information is more evident when we look into the contact of the responsible for the domain names. In Figure 3(d) we observe that more than 40% of Booters (17/42) have their contact hide from the WHOIS information. Usually the contact is hidden by many services, among them: "WhoisGuard Protected" (WGP - 7/42), "Whois Privacy Protect" (WPP - 4/42), and "NameCheap" (NC - 2/42).

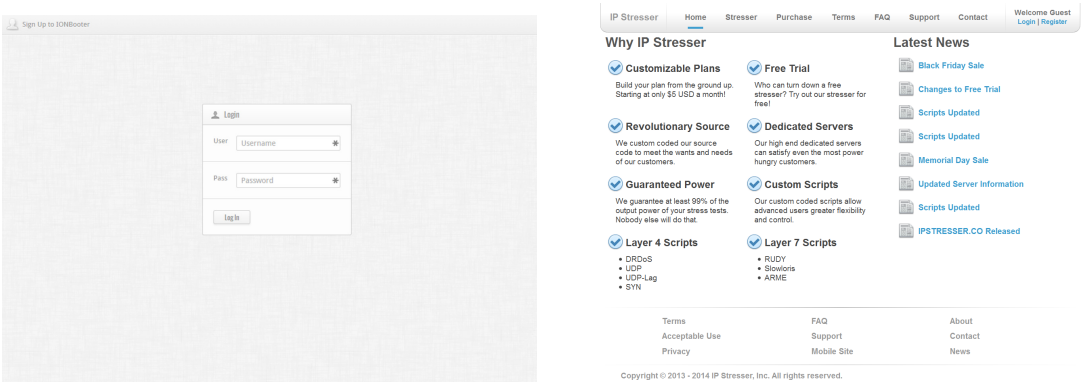
4.5. Website content analisys

The last feature that we use to characterize Booter websites is based on their content analysis. This analysis is divided in two parts: (i) the visual interface and (ii) the meta data, that is: the description and the keywords used to define Booter websites.

4.5.1. Visual interface

By manually accessing the 42 Booters we notice that they have only two completely different types of first page: or (i) a very simple first page containing a simple login interface, for example showed in Figure 4(a), or (ii) a verbose page full of textual content and appealing advertisements, for example shown in Figure 4(b). For both types Booters provide a very simple and user-friendly interface. The most remarkable finding related

to the visual interface was that while more than a half of Booters (22/42) has a “login” button in their main page only 1 non-Booter has such button.



(a) Booter F - login page.

(b) Booter E - login page.

Figure 4. Examples of login pages.

4.5.2. Meta data

When we look into meta data used to define Booters, almost 62% (26/42) has neither description nor keywords, showed in Figure 5. This result is very similar and consistent with the simplicity of Booters visual interface (described in the previous section). We also observe that there is not a clear distinction between non-Booters that use meta data. Therefore, although the meta data can help on the understanding the purpose of a website, because of insufficient information, we can not use this feature to define if a website is a Booter or not.

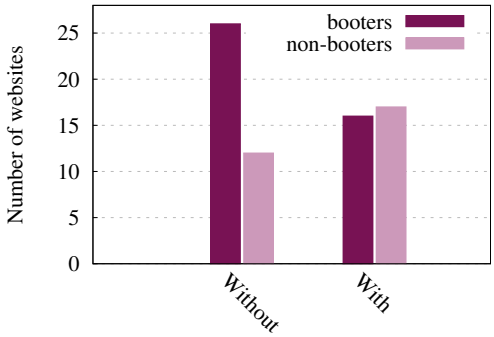


Figure 5. Meta data.

Note that in theory meta data is fundamental to have a better positioning in search engines, such as Google Search. Therefore we decide to correlate the popularity of a Booter with the existence or not of meta data. To do so we use the Alexa’s Rank [7] to get information about the 42 Booter webpages. By doing so, in general we observed that Booters with meta data have a higher ranking, for example in Table 6 two Booters are

[7] http://www.alexa.com

shown. Booter E that has meta data was ranked around the 273k position, while Booter F had a thousand times lower position. Note that the highest Alexa's Rank value is 1 (that usually has Google or Facebook).

Name	Meta data		Alexa's Rank
	Description	**Keywords**	
Booter E	Powerful and Affordable Stress Testing	stresser, denial of service, dos, ddos, dr-dos, syn, ssyn, udp, sudp, udp-lag, rudy, slowloris, arme	272.979
Booter F	-None-	-None-	2.580.782

Table 6. Examples of Booter meta data and their popularity

We are aware that meta data is not the only determining factor to have a higher ranking in the Internet. However, we are positively surprised to observe a clear relation between both factors.

4.6. A summary of the representative Booter features and a brief validation

The goal of this section is to summarize the most relevant features of Booters, which are based on what did we learn from our analysis. In addition we perform a brief validation to show that those features can be used to classify Booters.

In Section 4.2 we conclude that URL type 1, i.e. those that are composed by only the URL hostname, are more suitable to be a Booter. In Section 4.3 we notice that all Booters have less than 50 subpages and in general 2 levels of depth. We also observed that websites, which have pages "register" and "tos" are more often Booters. When it comes to WHOIS information, in Section 4.4, we highlight that Booters tend to use services from DDoS protection companies, such as CloudFlare or HyperFilter. The domain names used by Booters are most likely registered in 2012 or later, and the used registrar is most likely Enom. Moreover, information about the owners of Booters is hidden using services such as "WhoisGuard". Finally, in Section 4.5.1, we showed that Booters often have a simple interface with a login button. Through those observations we define a list with the main features that has a higher change to classify generic URLs as Booter websites:

1. Number of pages less than 50.
2. Depth level of the website of maximum 2.
3. Presence of registration page.
4. Presence of terms of service page.
5. Domain creation time 2012 and later.
6. Obfuscated WHOIS data.
7. Protected by a DPS.
8. Specific registrar: Enom.
9. Login button on page.

We decide to analyse if indeed the list of features can be used to classify Booters. To do so, we collected 3248 URLs related to Booters using the the search terms (Sec. 3.2) and our crawler (Sec. 3.3), on 30th November 2014. After filtering URLs based on URL type 1, we found 156 to perform a manual classification. From the manual classification we found 87 Booters websites and 69 as non-Booters. Then, for each one of the 156 URLs we count how many of the 9 features they have. The results are shown in Figure 6

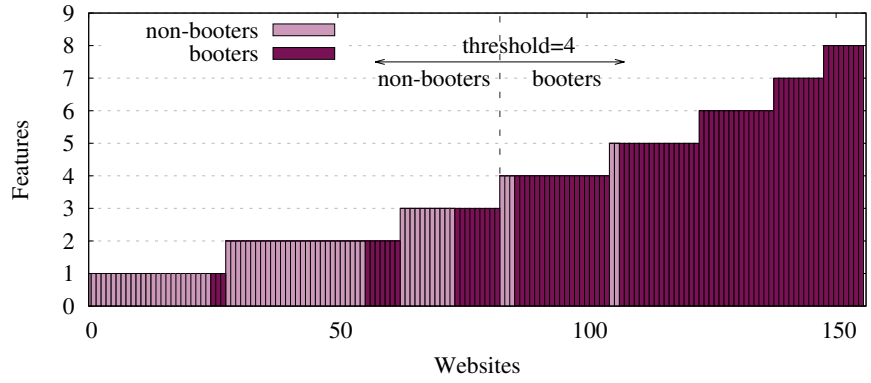


Figure 6. Representative features related to Booters per websites.

Table 7. Accuracy of threshold used to classify Booters

Threshold	True Positive	True Negative	Accuracy	False Positive	False Negative
3	78	52	**83%**	16	10
4	**69**	**63**	**85%**	**5**	**19**
5	50	66	**74%**	2	38
6	33	68	**65%**	0	55

The results in Figure 6 show clearly that the more features related to Booters a website has, the higher is the chance that such website is a Booter. Then, in the future, those features indeed can be used to automate the classification of Booters. However, the only problem that future researchers will have to determine where the threshold of features should be placed. Table 7 show the the accuracy of the threshold of our analysis.

Based on the retrieved URLs and our 9 features we found that URLs that have 4 or more features (threshold 4) show better results in terms of accuracy ((true negative +true positive)/population), 85%. Although this result is very promising, further research should be done to increase the accuracy.

5. Conclusion and Future work

By aiming to help security specialists to retrieve a comprehensive list of Booters, we performed a thorough characterization of Booter websites. First, we discovered the most common features to analyse websites. Then, we developed a crawler to retrieve a representative set of URLs and additional information to perform analysis. After an extensive analysis we highlight the 9 main characteristics of Booter websites. Finally, by using those characteristics we demonstrate that a list of Booters can be retrieved and classified with 85% of accuracy.

We conclude that although the 9 features are representative in the classification of Booter websites, more research needs to be done to achieve a fully automated methodology to retrieve a comprehensive list of Booters. As a future work, we aim to extend the number of retrieved URLs by adding additional sources of information, such as hacker forums, twitter, and youtube. Moreover, the analysis should include URLs type 3 and 4.

More research should be done to improve accuracy, for example by in depth analysis of the meta data or by assigning weights to the features. Since we expect Booters to

evolve, we consider further investigation of the analysis using machine learning. Moreover we urge for investigating the legal aspects of what is allowed as stress testing, including definition of procedures for authentication and verification of identity of people using these services.

Acknowledgments

This work was funded by the Network of Excellence project FLAMINGO (ICT-318488), which is supported by the European Commission under its Seventh Framework Programme.

References

[Akamai Technologies 2014] Akamai Technologies (2014). Prolexic Quarterly Global DDoS Attack Report (Q1 2014). http://www.prolexic.com/knowledge-center-ddos-attack-report-2014-q1.html.

[Arbor Networks 2014] Arbor Networks (2014). The Risk vs. Cost of Enterprise DDoS Protection. http://pages.arbornetworks.com/WebsiteRiskvsCost_Ent_DDoS_Protection.html.

[Hammami et al. 2006] Hammami, M., Chahir, Y., and Chen, L. (2006). Webguard: A web filtering engine combining textual, structural, and visual content-based analysis. *Knowledge and Data Engineering, IEEE Transactions on*, 18(2):272–284.

[Jo et al. 2013] Jo, I., Jung, E., and Yeom, H. Y. (2013). Interactive website filter for safe web browsing. *Journal of Information Science and Engineering*, 29(1):115–131.

[Karami and McCoy 2013] Karami, M. and McCoy, D. (2013). Understanding the Emerging Threat of DDoS-as-a-Service. In *6th USENIX Workshop on Large-Scale Exploits and Emergent Threats*. USENIX.

[Kassner 2013] Kassner, M. (2013). What's better than creating your own DDoS? renting one. http://www.techrepublic.com/blog/it-security/whats-better-than-creating-your-own-ddos-renting-one/.

[Kovacevic et al. 2004] Kovacevic, M., Diligenti, M., Gori, M., and Milutinovic, V. (2004). Visual adjacency multigraphs-a novel approach for a web page classification. In *Proceedings of SAWM04 workshop, ECML2004*.

[Krebs 2013a] Krebs, B. (2013a). Ragebooter: 'Legit' DDoS Service, or Fed Backdoor? http://krebsonsecurity.com/2013/05/ragebooter-legit-ddos-service-or-fed-backdoor/.

[Krebs 2013b] Krebs, B. (2013b). The world has no room for cowards. http://krebsonsecurity.com/2013/03/the-world-has-no-room-for-cowards/.

[Lindemann and Littig 2006] Lindemann, C. and Littig, L. (2006). Coarse-grained classification of web sites by their structural properties. In *Proceedings of the 8th annual ACM international workshop on Web information and data management*, pages 35–42.

[Lindemann and Littig 2007] Lindemann, C. and Littig, L. (2007). Classifying web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 1143–1144.

[Ma et al. 2009] Ma, J., Saul, L. K., Savage, S., and Voelker, G. M. (2009). Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254.

[Mirkovic and Reiher 2004] Mirkovic, J. and Reiher, P. (2004). A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53.

[Musthaler 2012] Musthaler, L. (2012). DDoS-as-a-Service? You Betcha! It's Cheap, It's Easy, and It's Available to Anyone. `http://www.securitybistro.com/?p=4121`.

[Ponemon Institute 2014] Ponemon Institute (2014). Cyber Security on the Offense: A Study of IT Security Experts. `http://security.radware.com/uploadedFiles/Resources_and_Content/Attack_Tools/CyberSecurityontheOffense.pdf`.

[Rajalakshmi and Aravindan 2011] Rajalakshmi, R. and Aravindan, C. (2011). Naive bayes approach for website classification. In *Information Technology and Mobile Communication*, pages 323–326. Springer.

[Rouse 2007] Rouse, M. (2007). Stress testing. `http://searchsoftwarequality.techtarget.com/definition/stress-testing`.

[Safe Keys 2013] Safe Keys (2013). Top 10 DDoSer's (Booters, Stressers. `http://www.safeskyhacks.com/Forums/showthread.php?39-Top-10-DDoser-s-(Booters-Stressers)`.

[Santanna et al. 2015a] Santanna, J. J., Durban, R., Sperotto, A., and Pras, A. (2015a). Inside booters: an analysis on operational databases. In *14th IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*. <to appear>.

[Santanna and Sperotto 2014] Santanna, J. J. and Sperotto, A. (2014). Characterizing and mitigating the ddos-as-a-service phenomenon. In *8th IFIP International Conference on Autonomous Infrastructure, Management, and Security (AIMS 2014)*, pages 74–78.

[Santanna et al. 2015b] Santanna, J. J., van Rijswijk-Deij, R., Sperotto, A., Hofstede, R., Wierbosch, M., Granville, L. Z., and Pras, A. (2015b). Booters - an analysis of ddos-as-a-service attacks. In *14th IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*. <to appear>.

S-Trace: Construindo Caminhos Causais em Redes Definidas por Software

Fabrício B. de Carvalho[1], **Ronaldo A. Ferreira**[1]

[1]Faculdade de Computação (FACOM)
Universidade Federal de Mato Grosso do Sul (UFMS)
CEP – 79070-900 – Campo Grande – MS – Brasil

{fabricio,raf}@facom.ufms.br

Abstract. *Determining hardware and software elements used to process an application request and group them in a set, called causal path, that exposes relevant parameters, such as processing time and delays, and that may explain the behavior of the application is a challenging task. Several tools for building causal paths have been proposed for the current Internet architecture, but none so far explores features offered by Software Defined Networks (SDN). This paper proposes S-Trace, a tool for building causal paths that does not modify applications and that uses specific features of SDN to build precise causal paths. To build a causal path, S-Trace intercepts library function calls to correlate communication events between processes and uses record and replay techniques in SDN to correlate network events. S-Trace was evaluated using a benchmark (TPC-W) and an application that emulates the behavior of multi-tier applications that was instrumented to validate the causal paths constructed by S-Trace. The experimental results show that S-Trace builds correct causal paths at the cost of a small overhead associated with some library function calls.*

Resumo. *Determinar corretamente elementos de hardware e software utilizados no processamento de requisições de uma aplicação e agrupá-los em um conjunto, denominado caminho causal, que exponha parâmetros relevantes, como tempo de processamento e atrasos, e que possam explicar o comportamento da aplicação é uma tarefa extremamente desafiadora. Diversas ferramentas de construção de caminhos causais foram concebidas para a arquitetura atual da Internet, mas nenhuma delas até então explora funcionalidades oferecidas pelas redes definidas por software (SDN* - Software Defined Networks*). Este trabalho propõe S-Trace, uma ferramenta de construção de caminhos causais que não exige instrumentação de aplicações e que utiliza funcionalidades específicas de SDN para construir caminhos causais precisos. Para construir um caminho causal, S-Trace intercepta chamadas de função de bibliotecas para correlacionar eventos de comunicação entre processos e utiliza técnicas de gravação e reprodução de tráfego de SDN para correlacionar eventos de rede. S-Trace foi avaliada utilizando um benchmark (TPC-W) e uma aplicação que emula o comportamento de aplicações multicamadas e que foi totalmente instrumentada para validar os caminhos causais gerados. Os resultados experimentais mostram que S-Trace gera caminhos causais corretos ao custo de um pequeno overhead em algumas chamadas de função de biblioteca.*

1. Introdução

Centros de processamento de dados modernos hospedam milhares de servidores para prover serviços de Internet de larga escala, sendo que cada servidor, normalmente, executa um conjunto diverso de aplicações de rede. Grandes empresas, como a HP (*Hewlett-Packard*), possuem mais de 6000 diferentes aplicações em uso em um único centro de dados [Scheck 2008]. As interações nesse ambiente — sejam entre aplicações ou entre dispositivos ou entre aplicações e dispositivos — se tornam extremamente complexas à medida que novos dispositivos ou aplicações são adicionadas ao ambiente. Balanceadores de carga, servidores proxy e dispositivos de NAT são fatores complicadores para se determinar e compreender as interações entre as aplicações, pois modificam o que seria o fluxo normal de uma conexão segundo o princípio fim-a-fim [Saltzer et al. 1984].

As interações entre as aplicações podem revelar dependências tanto diretas quanto indiretas. Um exemplo simples é uma requisição de um navegador a um servidor *Web*. Existe uma dependência direta entre a requisição do cliente e a resposta do servidor, mas podem haver também várias dependências indiretas, como: resolução de nomes via DNS, tradução de endereço por dispositivos NAT ou servidores de *proxy*, acesso a um servidor de banco de dados ou de autenticação pelo servidor *Web*, etc. Além das dependências de serviços e aplicações, há várias dependências de elementos de hardware, como *switches*, roteadores e enlaces.

Os elementos envolvidos no processamento de uma requisição, tanto de hardware como de software, podem ser agrupados em um conjunto denominado caminho causal. Em situações de falhas ou instabilidades, um caminho causal bem definido auxilia na identificação dos componentes que estão apresentando problemas. Várias ferramentas, entre elas Sherlock [Bahl et al. 2007], X-Trace [Fonseca et al. 2007] e Nemo [Júnior et al. 2013], utilizam o conceito de caminho causal para determinar qual elemento do conjunto está afetando o processamento da requisição. Essas ferramentas são divididas em dois grandes grupos: as intrusivas e as não intrusivas. A principal diferença entre os grupos é que nas intrusivas as aplicações precisam ser modificadas para se inserir informações de controle, enquanto nas não intrusivas não há essa necessidade.

A maioria das ferramentas de construção de caminhos causais até então conhecidas foram concebidas para a arquitetura atual da Internet. Essa arquitetura tem sido fortemente criticada por ser de difícil alteração e evolução, sendo inclusive rotulada de ossificada. O novo paradigma de *Rede Definida por Software* (SDN – *Software Defined Network*) foi proposto para contornar os problemas da arquitetura atual da Internet. SDN proporciona a dissociação entre o plano de controle e o plano de dados, permitindo que elementos externos de software exerçam funções do plano de controle e alterem o comportamento do plano de dados. A partir dessa dissociação, a arquitetura das SDNs permite que a inteligência da rede seja concentrada lógicamente em um único ponto. Esse ponto possui uma visão global da rede e simplifica significativamente o gerenciamento da rede.

Este trabalho propõe S-Trace, uma ferramenta não intrusiva de construção de caminhos causais que explora aspectos positivos das ferramentas de construção de caminhos causais intrusivas e não intrusivas, além da separação de planos fornecida por SDN. Para construir um caminho causal, S-Trace intercepta

chamadas de função de bibliotecas para correlacionar os eventos de comunicação inter-processo (IPC – *Inter-Process Communication*) utilizados ao processar uma requisição. S-Trace também utiliza técnicas de gravação e reprodução de tráfego de SDN para correlacionar eventos de rede. Os pacotes de dados são capturados e enviados para um sistema de armazenamento (*data store*) para reprodução posterior em um ambiente virtualizado. A reprodução do tráfego de rede é utilizada para aprimorar caminhos causais construídos com informações apenas dos extremos das conexões, pois permite correlacionar eventos de rede que possam alterar informações dos pacotes, como, por exemplo, tradução de endereços por dispostivos balanceadores de carga ou NAT.

Para a avaliação de S-Trace, foram utilizados um *benchmark* e uma aplicação representativa dos principais modelos de implementação de aplicações distribuídas. O *benchmark* TPC-W [Menasce 2002] foi utilizado para emular aplicações de comércio eletrônico (*e-Commerce*). A aplicação emula o comportamento de uma aplicação multicamadas (*multi-tier*) que realiza comunicações entre múltiplos processos e *threads* em diversos níveis de profundidade. Ela utiliza diferentes mecanismos de comunicação entre processos e foi instrumentada para mostrar o caminho causal exato de uma requisição. O objetivo principal dessa aplicação é comparar os resultados gerados por S-Trace com os caminhos causais exatos gerados pela instrumentação da aplicação. Os resultados da avaliação mostram que S-Trace é capaz de construir caminhos causais precisos, mesmo na presença de falhas e de dispositivos NAT que modificam os endereços IP das conexões utilizadas nas requisições, fatores que dificultam significativamente a construção de caminhos causais.

O restante deste trabalho está organizado como segue. A Seção 2 apresenta os trabalhos relacionados. S-Trace é descrita em detalhes na Seção 3. A Seção 4 demonstra os resultados dos experimentos e o impacto da utilização de S-Trace, e a Seção 5 conclui este trabalho.

2. Trabalhos Relacionados

Várias ferramentas foram propostas na literatura para construção de caminhos causais como mecanismo auxiliar de detecção de falhas e sobrecargas, mas todas elas foram concebidas para a arquitetura atual da Internet e não exploram possíveis funcionalidades fornecidas pela separação dos planos de dados e de controle em SDN. Uma das principais características de S-Trace é a utilização de serviços de gravação e reprodução de tráfego em SDN para correlação de eventos de rede. Recentemente, a ferramenta OFRewind [Wundsam et al. 2011] utilizou esses serviços em uma implementação relativamente simples devido à presença de um controlador logicamente centralizado. Entretanto, eles são de difícil implementação na arquitetura atual da Internet, principalmente pela natureza distribuída dos protocolos de roteamento.

Ferramentas intrusivas, como Pinpoint [Chen et al. 2004] e X-Trace [Fonseca et al. 2007], modificam as aplicações para que identificadores únicos sejam transportados em todo o processamento de uma requisição e para garantir a identificação de todos eventos que sejam provenientes de uma mesma requisição. X-Trace se destaca por ser a ferramenta mais intrusiva da literatura, pois, além de instrumentar aplicações, instrumenta toda a pilha de protocolo das estações de trabalho e servidores para prover visibilidade total no processamente de requisições. PIP [Reynolds et al. 2006], por outro lado, é uma ferramenta que, ao invés de modificar

aplicações, instrumenta *frameworks* para observar recursos utilizados pelas aplicações e suportar aplicações legadas sem a necessidade de modificações. vPath [Tak et al. 2009] expandiu o conceito de instrumentação para ambientes virtualizados para capturar comunicações entre *threads* de uma mesma máquina virtual.

Outro grupo importante de ferramentas são as não intrusivas que não modificam aplicações, protocolos ou *frameworks* para construir caminhos causais. BorderPatrol [Koskinen and Jannotti 2008], por exemplo, monitora as mensagens de aplicação trocadas entre processos e utiliza processadores de protocolo específicos para analisar as mensagens e correlacionar eventos que permitam a construção de caminhos causais. Sherlock [Bahl et al. 2007], por outro lado, modela as interações em um ambiente distribuído utilizando Redes Bayesianas e aplica tempo de coocorrência para determinar a relação entre eventos e construir os caminhos. Nemo [Júnior et al. 2013] reduz o tempo de inferência de Sherlock explorando conhecimento específico do domínio do problema e uma propriedade teórica de Redes Bayesianas.

Magpie [Barham et al. 2004] é a ferramenta que possui maior similaridade com S-Trace. Ambas monitoram as mensagens de aplicação e, a partir de esquemas de eventos, realizam as combinações necessárias para correlacionar os eventos e construir os caminhos causais. Contudo, o esquema de eventos de S-Trace é realizado de forma automática, diferentemente de Magpie em que administradores devem informar manualmente o esquema de cada aplicação. Além disso, S-Trace associa cabeçalhos dos pacotes com eventos de E/S para obter caminhos causais com granularidade fina, semelhante à X-Trace, mas sem realizar instrumentação na pilha de protocolo e nem nas aplicações.

3. S-Trace

A ferramenta S-Trace foi desenvolvida com intuito de construir e visualizar caminhos causais de requisições de aplicações distribuídas. Ela explora informações provenientes de uma SDN para identificar os componentes de hardware e software utilizados no processamento de uma requisição e para corrigir e aprimorar caminhos causais construídos com as informações dos extremos das conexões. S-Trace utiliza os conceitos de agente e gerente e seu fluxo geral está ilustrado na Figura 1.

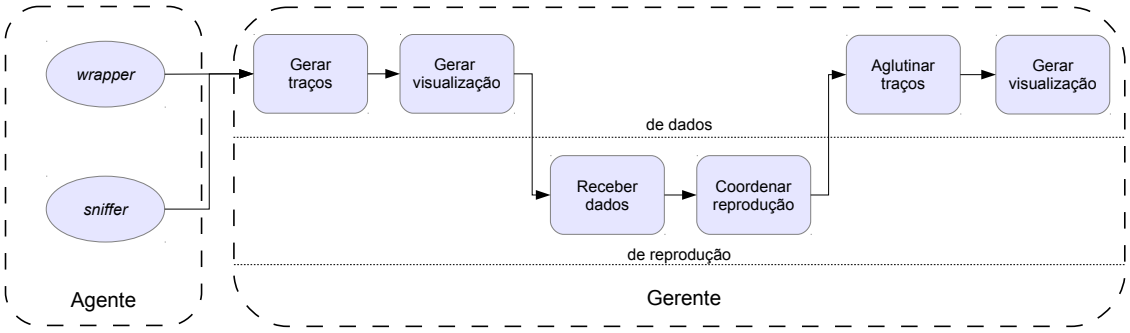


Figura 1. Fluxo geral da ferramenta S-Trace.

Os agentes, que são instalados em máquinas a serem monitoradas (estações de trabalho ou servidores), são divididos em dois componentes: *wrapper* e *sniffer*. O *wrapper* monitora as chamadas de função utilizadas durante a execução das aplicações e

o *sniffer* captura pacotes enviados e recebidos pelas estações monitoradas. Ambos enviam informações do monitoramento para o gerente de dados por meio de canais confiáveis e exclusivos, utilizando o protocolo TCP.

O gerente da ferramenta S-Trace é dividido em gerente de dados e gerente de reprodução. O gerente de dados é responsável por receber informações enviadas pelos agentes, construir — em um estágio inicial — os caminhos causais e gerar um arquivo com os dados para a visualização dos caminhos. O gerente de reprodução é responsável por armazenar e reproduzir o comportamento da rede original e fornecer informações ao gerente de dados para refinamento dos caminhos causais. A reprodução do comportamento da rede utiliza as mesmas técnicas de gravação e reprodução do tráfego de rede da ferramenta OFRewind [Wundsam et al. 2011].

3.1. Componente *wrapper*

O componente *wrapper* é encapsulado como uma biblioteca compartilhada, denominada *libwrapper*, para monitorar chamadas de função que as aplicações realizam durante a execução, capturar informações relevantes desse monitoramento e enviar essas informações para o gerente de dados. As principais informações são: LWPID, PID, PPID, argumentos da função, caminho absoluto de arquivos e carimbo de tempo. O *wrapper*, na realidade, realiza uma interposição de biblioteca e intercepta as chamadas para que funções da *libwrapper* sejam invocadas ao invés das funções da biblioteca original, sem que haja necessidade de instrumentação das aplicações. A interposição de biblioteca faz com que a biblioteca compartilhada seja carregada antes de qualquer outra. Para que isso aconteça, o carregador do sistema operacional deve ser informado da existência da *libwrapper* e deve carregá-la junto com a aplicação que será monitorada. Este último passo é facil de ser realizado, pois basta que a variável de ambiente `LD_PRELOAD` seja inicializada com o caminho completo da *libwrapper*.

As funções da *libwrapper* são apenas invólucros que realizam pequenos processamentos e invocam as funções originais. Com isso, o comportamento das funções e, consequentemente, das aplicações, não é alterado pelo componente *wrapper*. As funções da *libwrapper* que foram implementadas neste trabalho estão listadas na Tabela 1.

Tabela 1. Funções da biblioteca *libwrapper*.

`read`	`recv`	`write`	`send`
`socket`	`bind`	`listen`	`accept`
`connect`	`close`	`fopen`	`fclose`
`dup`	`dup2`	`fork`	`msgget`
`msgsnd`	`msgrcv`	`shmget`	`shmat`
`semget`	`semop`	`pipe`	`pipe2`
`mkfifo`	`readv`	`writev`	`sendfile`

As funções implementadas pelo componente *wrapper* possuem fluxo predefinido, que é dividido em três fases. A primeira fase consiste em invocar a função original da biblioteca correta com os mesmos parâmetros, armazenar o valor de retorno da função original e retorná-lo pela função da *libwrapper*. Dessa forma, o comportamento da aplicação não é modificado. A segunda fase consiste em extrair informações referentes à aplicação – como identificadores de *threads*, de processos e de descritores de mecanismos

de IPC – que auxiliam S-Trace a construir os caminhos causais das requisições. A última fase consiste em enviar essas informações para o gerente de dados, que será descrito na Seção 3.3.

O componente *wrapper* utiliza processadores de protocolo para interpretar as mensagens trocadas entre as aplicações e extrair informações como delimitadores de mensagens de aplicação e atributos específicos de protocolo. O conhecimento necessário para o desenvolvimento de processadores é específico de protocolo e não de aplicação. Neste trabalho, foram utilizados processadores dos protocolos HTTP e MySQL. Como exemplo de atributo específico de protocolo, pode-se citar o caso em que cliente e servidor decidem manter conexões persistentes. O *wrapper* também extrai informações da função que está sendo executada como: valor de retorno; LWPID, PID e PPID da *thread* que invocou a função; descritor de arquivo; etc. Com essas informações, o gerente de dados produz caminhos causais iniciais que serão aprimorados posteriormente com informações fornecidas pelo gerente de reprodução.

A utilização do componente *wrapper* não modifica as aplicações e nem as funções originais. O componente modifica apenas o fluxo de dados das aplicações com um novo estágio adicionado ao caminho do fluxo. Esse estágio captura e envia informações sobre a aplicação e a função para o gerente de dados e em seguida o fluxo prossegue normalmente. A adição desse novo estágio não afeta significativamente o desempenho da aplicação, pois são operações rápidas e curtas. A Seção 4 avalia o *overhead* introduzido pelo *wrapper* em diferentes chamadas de função.

3.2. Componente *sniffer*

O componente *sniffer* monitora todos os pacotes enviados e recebidos pela estação de trabalho ou servidor e envia os cabeçalhos dos pacotes para o gerente de dados. Esse componente é inicializado como um *daemon* e não exige interação com os usuários. Apesar de capturar os pacotes, o *sniffer* não influencia no tráfego de rede, muito menos no comportamento das aplicações, pois ele realiza uma cópia dos pacotes para não alterar o fluxo normal da conexão.

Para realizar o monitoramento, o componente *sniffer* captura pacotes utilizando a biblioteca *libpcap*, disponível para os principais sistemas operacionais atuais. Para cada pacote recebido ou enviado pela estação monitorada, uma função de *callback* do *sniffer* é invocada pela *libpcap*.

As informações de tamanho dos cabeçalhos do pacote, o carimbo de tempo do momento em que o pacote trafegou pela estação de trabalho ou servidor e os cabeçalhos dos pacotes são encapsulados em uma estrutura de dados. O *sniffer* envia essas informações encapsuladas por meio de um canal exclusivo e confiável para o gerente de dados, utilizando o protocolo TCP. Contudo, esse envio é realizado, normalmente, por meio da mesma interface de rede que é monitorada pelo componente *sniffer*. Dessa forma, há necessidade de um filtro especial no monitoramento dos pacotes para que os pacotes com destino e origem ao gerente de dados não sejam capturados. Se esse filtro não fosse utilizado, haveria um laço infinito no monitoramento dos pacotes.

3.3. Gerente de Dados

O gerente de dados é um dos principais componentes de S-Trace. Ele recebe e organiza os dados enviados pelos agentes, gera caminhos causais iniciais, produz uma visualização

prévia dos caminhos, troca informações com o gerente de reprodução, aperfeiçoa os caminhos causais e, por fim, constrói os caminhos causais finais.

O gerente de dados cria duas listas, uma para os *wrappers* e outra para os *sniffers*. Os dados recebidos são adicionados em suas respectivas listas ordenados pelos carimbos de tempo. Periodicamente, o gerente de dados inicia o processo de construção de caminhos causais a partir da lista de nós *wrappers*. Inicialmente, esse processo busca por nós que representem início e fim de uma conexão, pois eles representam delimitadores em um caminho causal. O final da conexão pode ser obtido por meio de funções que representem término de um fluxo ou de nós que utilizaram o processador de protocolo que interpretou e delimitou as mensagens de aplicação. Os nós recebidos entre os nós inicial e final são analisados para se verificar se possuem alguma ligação com o caminho. Em caso positivo, eles são adicionados ao caminho.

As mensagens trocadas pelos agentes e gerente são do tipo assíncronas e diversas mensagens podem ser encaminhadas a qualquer momento, inclusive durante o processo de construção dos caminhos causais. Para que a construção dos caminhos não interfira no recebimento de dados, o gerente de dados possui dois ponteiros extras: um para a lista dos *wrappers* e outro para os *sniffers*. Quando o processo é iniciado, há uma troca de ponteiros para que as listas não sejam modificadas durante o processo de construção e para que os recebimentos não sejam afetados.

A função que verifica se um nó n possui algum tipo de ligação com um caminho causal C é definida da seguinte maneira. Primeiramente, o conjunto S de LWPID baseado nos nós de C é definido. Caso o valor do campo LWPID de n pertença a esse conjunto, n possui ligação com C. Caso contrário, a função de ligação verifica se n possui algum mecanismo de IPC com algum nó de C. Os tipos de IPC verificados são: arquivo, *pipe*, *socket*, semáforo, fila de mensagens e memória compartilhada. Basicamente, a verificação desses mecanismos busca por identificadores comuns entre os nós para realizar a correlação e, assim, adicioná-los ao caminho causal. A função é utilizada pelo processo de construção de caminhos causais e é invocada para todos os nós *wrappers* que não foram adicionados a nenhum caminho causal. S-Trace assume que *threads* que compartilham blocos de memória ou dados estão processando a mesma requisição, o que nem sempre é verdadeiro e pode gerar falsos positivos. Falsos positivos podem ser minimizados ou até eliminados por análise de correlação temporal dos eventos. Entretanto, esse assunto não será abordado neste trabalho e será objeto de trabalho futuro.

Após as uniões dos caminhos causais, o gerente de dados inicia o processo de associação entre as operações realizadas pelas aplicações (*wrapper*) e os cabeçalhos dos pacotes trafegados pelas estações de trabalho ou servidores (*sniffer*), realizando a união entre os nós dos caminhos causais e os cabeçalhos dos pacotes. Essa operação é realizada por meio dos números de sequência utilizados pelo protocolo TCP. A associação é dividida em duas partes, a primeira coleta todas operações de entrada e todos os cabeçalhos de entrada e a outra parte coleta os dados de saída. Essa divisão é realizada pois os nós *wrappers* estão ordenados pelo carimbo de tempo, garantindo a ordem de invocação das funções. Entretanto, o carimbo de tempo dos nós *sniffers* não é um forte parâmetro para realizar a associação, pois não há garantia da ordem de saída ou de entrada dos pacotes por meio do tempo.

Após o processo de construção dos caminhos causais, enriquecidos com as informações dos cabeçalhos capturados, o gerente de dados gera um arquivo de visualização do tipo `dot` que contém grafos que representam todos os caminhos causais construídos. Cada nó representa alguma chamada de função que a aplicação invocou. Caso a chamada produza pacotes de rede, novos nós são criados para representar os pacotes. Após a construção do arquivo `.dot`, o gerente percorre cada caminho procurando por inícios de conexão que não foram unidos durante o processo de construção dos caminhos causais. Se existirem nós de conexão que não foram unidos, esses nós são informados ao gerente de reprodução, para que, posteriormente, caminhos causais que aparentemente não possuem relação possam ser unidos. O intuito dessa notificação é aprimorar os caminhos causais construídos pelo gerente de dados com informações provenientes da reprodução do tráfego de rede, por exemplo.

Em um primeiro momento, a notificação dos inícios de conexão para o gerente de reprodução não possui nenhum motivo, pois se as informações fornecidas pelos agentes não forem suficientes, aparentemente não há ligação entre os caminhos. Entretanto, caso alguma técnica de modificação de campos do pacote seja utilizada pela rede, por exemplo, NAT, o gerente de dados e os agentes da ferramenta não saberiam dessa utilização. Porém, o gerente de reprodução possui essa informação que é fornecida pela centralização lógica da inteligência da rede fornecida por SDN. Essa informação é essencial para que o gerente de dados realize a ligação entre os caminhos causais. A Figura 2 ilustra um exemplo de dois caminhos causais aparentemente sem nenhum tipo de ligação, porém, há uma regra em um comutador que realiza a modificação dos endereços IP dos pacotes, fazendo com que os caminhos sejam ligados.

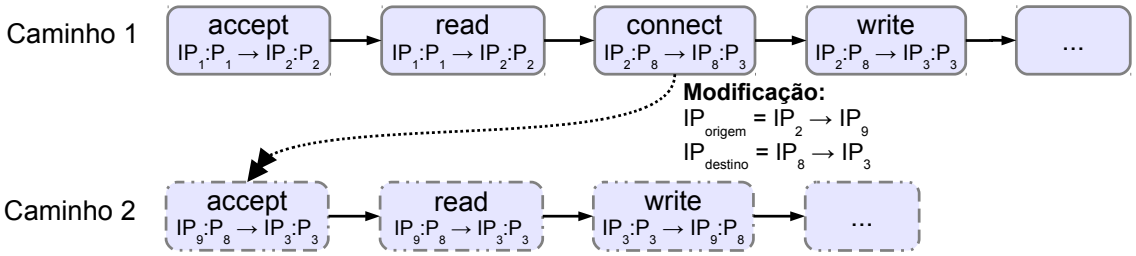


Figura 2. Exemplo de dois caminhos causais sem ligação aparente entre os caminhos. Durante o tráfego, modificações nos endereços IP dos pacotes são realizadas, apontando ligação entre os caminhos que não é observada pelos extremos das conexões.

A simples utilização da técnica NAT faz com que um único caminho causal seja particionado, erroneamente, em duas requisições diferentes. Esse fato acontece pois não há nenhum tipo de união, nesse estágio, entre os caminhos. Entretanto, com a utilização do gerente de reprodução, essa situação é contornada, e os caminhos causais são unidos. Observando a Figura 2, sem a informação das modificações, os Caminhos 1 e 2 são totalmente distintos, o que é uma falsa dedução. As informações de modificação não são visíveis para os agentes e para o gerente de dados, apenas o gerente de reprodução possui essa informação, pois ele tem acesso às regras de fluxo instaladas nos comutadores de rede.

O gerente de reprodução, após receber os inícios de conexão, realiza diversos

procedimentos (que são explicados detalhadamente na próxima seção) e envia ao gerente de dados informações de instalações de regras nos comutadores de rede referentes aos dados recebidos. Essas informações são utilizadas pelo gerente de dados para realizar uniões entre os caminhos causais que, aparentemente, não possuiam nenhum tipo de ligação. Dessa forma, o gerente de dados novamente invoca o processo de união entre os caminhos causais juntamente com os dados recebidos pelo gerente de reprodução. Feito isso, o processo de visualização também é invocado novamente, gerando um novo arquivo `.dot` final, com os caminhos causais unidos e bem definidos.

3.3.1. Gerente de Reprodução

O gerente de reprodução possui o objetivo de gravar o trafégo de rede de modo escalável, consistente e controlado, receber informações sobre os caminhos causais fornecidos pelo gerente de dados, coordenar a reprodução do tráfego armazenado e informar ao gerente de dados as regras de fluxo que influenciam nos caminhos causais e que foram instaladas nos comutadores OpenFlow. O gerente intercepta as regras de fluxo, observando todas as ações tomadas pelos comutadores em determinados fluxos.

O núcleo do gerente de reprodução é extraído da ferramenta OFRewind [Wundsam et al. 2011] que consolida as técnicas de gravação e reprodução de tráfego de rede em um ambiente programável utilizando o protocolo OpenFlow. O gerente de reprodução explora essas técnicas para notificar qual foi o comportamento da rede durante o processo de gravação, ou seja, quais foram as ações tomadas pelos comutadores de rede.

O gerente de reprodução conta com dois módulos: um é responsável por receber e enviar mensagens para o gerente de dados e o outro é responsável por interceptar regras de fluxo a serem instaladas nos comutadores. O módulo que recebe e envia dados para o gerente de dados deve receber os dados e os encapsular em estruturas de dados de tal forma que, o módulo de interceptação seja capaz de verificar se alguma regra de fluxo influencia nos dados recebidos. De forma semelhante ao gerente de dados, esse módulo possui uma lista para receber os dados e ordená-los pelo carimbo de tempo.

A atuação do módulo que intercepta as regras de fluxo não afeta o comportamento da rede, pois as regras não são alteradas, apenas informações de modificação de campos de pacotes são extraídas das regras. As regras observadas são apenas aquelas que possuem alguma ligação com os nós *wrappers* que indicam inícios de conexão que não foram unidos durante o processo de construção de caminhos causais do gerente de dados. Essa ligação é fornecida pelos endereços IP e portas de comunicação da regra e do nó *wrapper*.

O gerente de reprodução é um componente fundamental para que o gerente de dados produza caminhos causais confiáveis e bem definidos, pois a reprodução do comportamento da rede revela relacionamentos que não podem ser capturados nos extremos de uma conexão. Essa contribuição é realizada, principalmente, pela captura das ações das regras de fluxo instaladas nos comutadores, fazendo que modificações nos pacotes de redes sejam observadas e notificadas para o gerente de dados por meio da interceptação das mensagens do plano de controle. Além disso, como todas as regras são instaladas em um momento pós-gravação, o comportamento do tráfego de rede não é alterado. Essa é uma característica importante da ferramenta OFRewind que é mantida pelo gerente de reprodução.

A Figura 3 ilustra o caminho causal construído pela combinação dos dois caminhos ilustrados na Figura 2. O gerente de reprodução observa as regras de fluxo que indicam as modificações ilustradas e informa ao gerente de dados sobre as modificações. Ao receber essa informação, o gerente de dados invoca novamente o processo de construção e os dois caminhos são unidos, formando um único caminho causal.

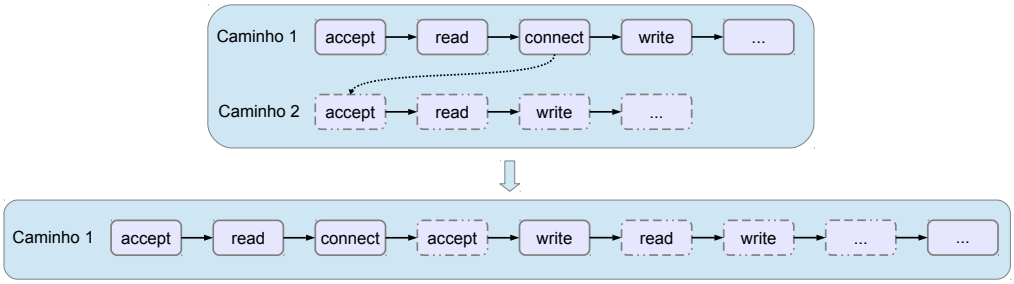


Figura 3. Exemplo de união entre caminhos causais. A linha tracejada indica que os dois nós possuem uma ligação que só foi possível identificá-la com a reprodução do tráfego de rede, observando as regras de fluxo.

4. Avaliação Experimental

S-Trace foi avaliada utilizando o *benchmark* (TPC-W) e uma aplicação (n-Tier) que emula o comportamento de aplicações multicamadas e que foi totalmente instrumentada para validar os caminhos causais gerados. A ferramenta Mininet [Lantz et al. 2010] foi utilizada para emular a rede, pois ela possibilita a criação de diferentes redes virtuais em um único computador, além de fornecer a rápida prototipação de diferentes redes virtuais baseadas no protocolo OpenFlow [McKeown et al. 2008]. A topologia de rede utilizada nesta avaliação foi composta por 13 *hosts* e 4 comutadores. Dentre os *hosts*, 8 são instâncias de n-Tier, 3 realizam requisições e 2 executam TPC-W.

O objetivo da avaliação experimental é mostrar que S-Trace captura de maneira confiável os elementos utilizados no processamento de uma requisição. Algumas falhas foram produzidas para demonstrar que os caminhos causais construídos são capazes de, além de auxiliar a compreender como as aplicações se comportam ao processar uma requisição, contribuir para detecção de falhas.

4.1. Aplicação n-Tier

A aplicação n-Tier foi desenvolvida para emular o comportamento de aplicações que utilizam múltiplas camadas, aspecto comum em grande parte das aplicações distribuídas usuais. n-Tier foi totalmente instrumentada para fornecer todos os eventos realizados pelas *threads* no processamento das requisições. Com isso, é possível comparar os caminhos causais construídos por S-Trace e os eventos fornecidos pela instrumentação. Cada instância de n-Tier é denominada réplica que possui um *pool* de *thread* e uma *thread* principal que recebe as requisições e as encaminha para uma das *threads* do *pool*, ou seja, a primeira camada de processamento.

A primeira camada de processamento, ao receber uma requisição, deve decidir se processa a requisição ou se realiza um encaminhamento. Caso a decisão seja não processar a requisição, então essa camada deve decidir se encaminha a requisição para

uma nova camada ou para uma outra instância da aplicação. Ambas decisões são determinadas aleatoriamente. A cada requisição, a *thread* decide, com base em uma variável aleatória com distribuição uniforme, se a requisição deve ser repassada ou não. A probabilidade de repasse é um parâmetro de execução da aplicação n-Tier.

O mecanismo de IPC `socket` é utilizado para o encaminhamento entre as réplicas, visto que, cada *host* produzido por Mininet possui um endereço IP próprio e executa uma instância da aplicação n-Tier. Os encaminhamentos entre as camadas de processamento podem ser realizados utilizando arquivo, *pipe*, fila de mensagens ou memória compartilhada.

Para mostrar a interação de S-Trace em uma SDN, a técnica de NAT foi implementada nos comutadores de tal forma que quando réplicas da aplicação n-Tier que estão na mesma sub-rede trocam informações entre si, as conexões são realizadas por meio dos endereços externos não pertencentes ao escopo de endereçamento da sub-rede. Dessa forma, quando os comutadores recebem pacotes de rede com esses endereços externos de destino, eles são instruídos a modificar o campo destino IP para o endereço da réplica correta. Além disso, o campo origem IP também é modificado para o endereço externo correto nos pacotes com as respostas das réplicas.

A Tabela 2 ilustra a quantidade de cada um dos mecanismos de IPC utilizado pela aplicação n-Tier em diferentes quantidades de requisições. O valor da somatória da quantidade de cada mecanismo de IPC é superior ao valor da quantidade de requisições porque vários mecanismos de IPC podem ser utilizados no processamento de uma única requisição.

Tabela 2. Quantidade de cada mecanismo de IPC utilizado pela aplicação n-Tier para processar diversas requisições.

	Mecanismos de IPC				
Requisições	Arquivo	*Pipe*	*Socket*	Fila de mensagens	Memória compartilhada
25	25	17	17	11	9
100	39	35	24	26	19
200	63	59	43	48	49
300	88	83	59	72	70
400	111	99	83	88	92
500	127	126	104	111	106
1000	218	239	228	243	199
10000	2012	2017	1634	2054	1964

4.2. TPC-W

TPC-W [Menasce 2002] é um *benchmark* para transações *Web* do tipo *e-Commerce* que utiliza o servidor *Web* Tomcat e o servidor de banco de dados MySQL. TPC-W realiza diversas requisições ao servidor *Web* para comparar o desempenho do servidor ao processar as requisições. As requisições são comuns a vários tipos de sistema *e-Commerce*, como: navegação pela lista de produtos, compra, venda, etc. Várias requisições são disparadas para simular a navegação em um sistema de comércio eletrônico.

Para ilustrar um dos benefícios de caminhos causais bem definidos, durante o experimento com TPC-W, o servidor de banco de dados foi desativado. Com o caminho

causal, foi possível observar que a *thread* que processa a requisição tenta iniciar conexão com o banco por diversas vezes sem sucesso. Com isso, também foi possível observar que o Tomcat não utilizou uma conexão persistente com o banco e, também, que uma única *thread* foi responsável por receber as conexões e repassar as requisições para as demais *threads*. Essas características são determinadas em arquivos de configuração na execução do servidor *Web*. Entretanto, com os caminhos construídos por S-Trace, é possível revelar esses aspectos sem examinar esses arquivos.

4.3. Resultados

Os caminhos causais construídos por S-Trace são ilustrados em forma de grafos em que cada nó representa uma função que foi invocada por uma *thread* de uma aplicação e também um pacote que trafegou na rede proveniente da chamada da função. Os nós em nível horizontal representam as chamadas de função e possuem as seguintes informações: endereço IP do agente que enviou os dados para o gerente, carimbo de tempo, identificação da função invocada, LWPID e PID. Os nós em nível vertical representam os pacotes de rede que trafegaram devido à função invocada. Estes nós possuem as seguintes informações: carimbo de tempo, *flags* do protocolo TCP, número de sequência do pacote e indicação de erro, caso exista.

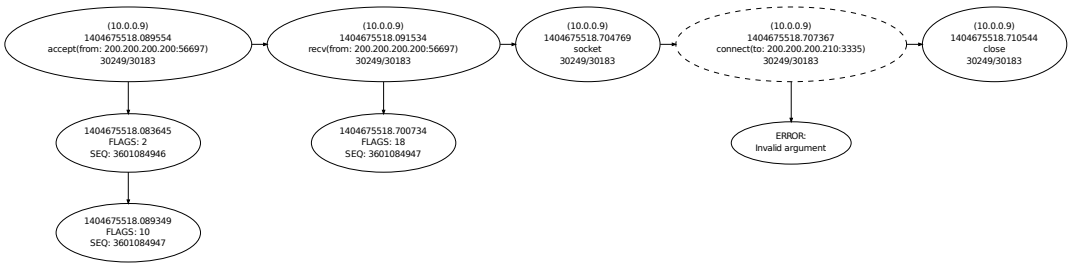


Figura 4. Caminho causal de uma requisição de n-Tier construído por S-Trace. Durante a execução, a conexão com destino `200.200.200.210:3335` não foi estabelecida e por isso, o nó foi contornado com linha tracejada indicando algum erro. Alguns nós do caminho foram omitidos para facilitar a visualização.

A Figura 4 ilustra um caminho causal construído por S-Trace. O tamanho do caminho foi reduzido para facilitar a visualização. Esse caminho causal é construído quando uma instância da aplicação n-Tier recebe a requisição e tenta conexão com a réplica com o endereço `200.200.200.210:3335`. Entretanto, essa conexão não foi estabecida retornando o erro `Invalid Argument`, neste caso, S-Trace faz com que este nó seja contornado com linha tracejada indicando o erro.

Foram realizadas 100 requisições para instâncias da aplicação n-Tier e em todas elas, S-Trace identificou todos os relacionamentos entre as *threads* e os mecanismos de IPC utilizados. Com base em uma variável aleatória com distribuição uniforme, uma réplica de n-Tier é escolhida como destino inicial para cada requisição. A comparação entre os caminhos causais construídos por S-Trace e os relacionamentos de n-Tier foi possível pois n-Tier foi instrumentada para fornecer os eventos realizados pelas *threads* no processamento das requisições.

O componente *wrapper* do agente da ferramenta S-Trace também foi avaliado para

demonstrar o impacto, em tempo de execução, da utilização do agente nas aplicações, mais especificamente, a utilização da biblioteca *libwrapper*. Para isso, uma aplicação cliente/servidor simples foi desenvolvida em que o cliente realiza uma conexão com o servidor, envia um *buffer* de mensagem e encerra a conexão. Além disso, durante a execução dessa aplicação, as funções `fork` e `fopen` foram invocadas.

Na avaliação de tempo de execução, é capturado o carimbo de tempo antes e depois da chamada das funções `send`, `fopen` e `fork`. O tamanho do *buffer* da mensagem a ser enviada variou entre 100, 1500 e 10000 bytes. Para cada tamanho de *buffer* utilizado na função `send` e para as funções `fork` e `fopen`, foram executadas 500 instâncias do cliente, produzindo 500 requisições. A Tabela 3 ilustra os resultados obtidos e os intervalos de confiança de 95% do tempo médio. Os resultados numéricos mostram que os atrasos inseridos pelo *wrapper* são desprezíveis.

Tabela 3. Tempo de execução médio de chamadas de função (ms).

	sem *wrapper* (I.C. 95%)	com *wrapper* (I.C. 95%)	*overhead*
send (100 bytes)	0,068±0,005	0,144±0,012	∼0,076
send (1500 bytes)	0,103±0,009	0,174±0,015	∼0,071
send (10000 bytes)	0,150±0,013	0,221±0,019	∼0,071
fopen	0,326±0,028	0,401±0,035	∼0,075
fork	0,477±0,041	0,553±0,048	∼0,076

Por limitações de espaço, esta seção apresenta um conjunto reduzido dos resultados obtidos com S-Trace. Outros experimentos, como: injeção de atrasos, réplica inexistente, encerramento do servidor de banco de dados, etc., foram realizados e podem ser consultados em [Carvalho 2014]. A dissertação [Carvalho 2014] também lista em seu apêndice a implementação das funções *wrappers* de S-Trace.

5. Conclusão

S-Trace é uma ferramenta de construção de caminhos causais que não exige instrumentação de aplicações, protocolos ou *frameworks* e que utiliza informações de fluxos de dados em um controlador SDN e de chamadas de funções de IPC da `libc` para construir caminhos causais. Os resultados experimentais são promissores, pois a ferramenta foi capaz de reconstruir caminhos causais precisos em execuções controladas de uma aplicação multicamadas que foi totalmente instrumentada. Entretanto, a ferramenta ainda está em um estágio inicial e mostra algumas direções para investigações e trabalhos futuros, como: análise de falsos positivos; implementação de novos processadores de protocolo de aplicação; avaliação do melhor local (*switch* ou *host*) para captura de pacotes; interface gráfica para visualização dos resultados; e informações que o controlador SDN pode fornecer para melhorar a precisão dos caminhos causais.

Referências

Bahl, P., Chandra, R., Greenberg, A., Kandula, S., Maltz, D. A., and Zhang, M. (2007). Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies. In *Proceedings of the 2007 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'07)*.

Barham, P., Donnelly, A., Isaacs, R., and Mortier, R. (2004). Using Magpie for Request Extraction and Workload Modelling. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation (OSDI'04)*.

Carvalho, F. B. (2014). Construção de Caminhos Causais em Redes Definidas por Software. Faculdade de Computação (FACOM), UFMS. `http://ndsg.facom.ufms.br/S-Trace.pdf`.

Chen, M. Y., Accardi, A., Kiciman, E., Lloyd, J., Patterson, D., Fox, A., and Brewer, E. (2004). Path-Based Failure and Evolution Management. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI'04)*.

Fonseca, R., Porter, G., Katz, R. H., Shenker, S., and Stoica, I. (2007). X-Trace: A Pervasive Network Tracing Framework. In *Proceedings of the 4th Symposium on Networked Systems Design and Implementation (NSDI'07)*.

Júnior, B. A. S., Carvalho, F. B., and Ferreira, R. A. (2013). Nemo: Procurando e Encontrando Anomalias em Aplicações Distribuídas. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'13)*.

Koskinen, E. and Jannotti, J. (2008). BorderPatrol: Isolating Events for Black-box Tracing. In *Proceedings of the 3rd EuroSys European Conference on Computer Systems (Eurosys'08)*.

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)*.

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*, pages 69–74.

Menasce, D. (2002). TPC-W: a Benchmark for *e-Commerce*. *Internet Computing, IEEE*, pages 83–87.

Reynolds, P., Edwin K., C., Wiener, J. L., Mogul, J. C., Shah, M. A., and Vahdat, A. (2006). PIP: Detecting the Unexpected in Distributed Systems. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI'06)*.

Saltzer, J. H., Reed, D. P., and Clark, D. D. (1984). End-to-end Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288.

Scheck, J. (2008). Taming Technology Sprawl. The Wall Street Journal. `http://online.wsj.com/article/SB120156419453723637.html`. Acessado em 13/12/2014.

Tak, B. C., Tang, C., Zhang, C., Govindan, S., Urgaonkar, B., and Chang, R. N. (2009). vPath: Precise Discovery of Request Processing Paths from Black-Box Observations of Thread and Network Activities. In *Proceedings of The 34th USENIX Annual Technical Conference (USENIX'09)*.

Wundsam, A., Levin, D., Seetharaman, S., and Feldmann, A. (2011). OFRewind: Enabling Record and Replay Troubleshooting for Networks. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference (USENIXATC'11)*.

Modelo analítico-determinístico para a priorização de tráfego em Redes Definidas por Software com Network Calculus

Michael Prieto Herńandez[1]*, Ricardo Girnis Tombi[1], Samuel Kopp[1]*, Daniel Batista[2], Cíntia Borges Margi[1], Regina Melo Silveira[1]

[1]Departamento de Engenharia de Computação e Sistemas Digitais – Escola Politécnica
Universidade de São Paulo

[2]Departamento de Ciência da Computação – Instituto de Matemática e Estatística
Universidade de São Paulo

{mhernandez,rtombi,samuel,cbmargi,regina}@larc.usp.br,

batista@ime.usp.br

Abstract. *Different classes of packets traffic require specific performance of network resources in terms of Quality of Service metrics. The models proposed to ensure end-to-end QoS have not been well implemented due to their complexity and lack of scalability. The Software Defined Networks feature an innovative paradigm that aims to make the networks programmable. This paradigm allows to know the state of the network and its topology in real time, which makes possible the reconfiguration of routes and allocating resources dynamically. This paper proposes an analytical-deterministic model that allows to prioritize IP packets traffic, taking advantage of SDN and basing the prioritization policies over definitions found in the Network Calculus mathematical theory. The results show the feasibility of using NetCalc as a tool in the packet flows policing in order to ensure QoS on a software defined network.*

Resumo. *Diferentes classes de tráfego requerem desempenho específico dos recursos de rede em termos de Qualidade de Serviço. Modelos propostos para garantir QoS fim-a-fim na literatura não foram implementados de forma satisfatória até hoje devido à sua complexidade e falta de escalabilidade. As Redes Definidas por Software (SDN) apresentam um paradigma inovador que torna as redes programáveis, permitindo conhecer estado e topologia das mesmas em tempo real, além de possibilitar configuração de rotas e alocação de recursos dinamicamente. Este trabalho apresenta um modelo analítico-determinístico para priorização de tráfego aproveitando as vantagens da SDN, baseado em políticas de priorização derivadas da teoria do Network Calculus. Os resultados obtidos comprovam a viabilidade do NetCalc para subsidiar controle de fluxos garantindo QoS em SDN.*

*Esse projeto foi financiado pelo processo nº 2014/50386-5, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)

1. Introdução

A distribuição de vídeo através da Internet tem se tornado cada vez mais frequente, e algumas previsões indicam que até o final desta década noventa porcento do tráfego da Internet será constituído por fluxos de vídeo [Cisco 2014]. Considerando as previsões de expansão de consumo desta mídia é necessário desenvolver mecanismos nas redes que permitam viabilizar soluções que apresentem bom desempenho, como robustez e escalabilidade.

Visando garantir QoS fim-a-fim, a Internet Engineering Task Force (IETF) propôs dois modelos, Serviços Integrados (IntServ) [Braden et al. 1994] e Serviços diferenciados (DiffServ) [Blake et al. 1998] os quais não foram satisfatoriamente implementados globalmente até hoje devido à sua complexidade e falta de escalabilidade [Ni 1999]. Embora a IETF tenha criado um mecanismo de transporte (Multi Protocol Label Switching) [Rosen et al. 2001] que possibilita a associação de requisitos de QoS, baseado nos rótulos carregados pelos pacotes, este carece de reconfiguração e adaptabilidade em tempo real [Egilmez and Dane 2012].

Devido às limitações da Internet atual em prover conexão com QoS e, consequentemente, em prover qualidade de experiência (QoE) compatível com as expectativas do usuário, os mecanismos de QoS continuam com grande foco, principalmente em se tratando de mecanismos que atendam as necessidades de sistemas autônomos como as redes dos Provedores de Serviços de Internet (ISP).

Com o surgimento das Redes Definidas por Software (SDN) [ONF 2012], é possível especificar a partir da implementação por linguagem de programação o comportamento de um comutador de pacotes ou *switch*, o que diminui custos e ao mesmo tempo, aumenta a flexibilidade do processo de encaminhamento de pacotes IP. Esta abordagem permite criar mecanismos, que através do monitoramento da rede, forneçam um controle ativo da mesma para superar problemas de limitações de recursos e congestionamentos.

Uma das implementações de maior repercussão de SDN é o protocolo OpenFlow [Mckeown et al. 2008], mantido atualmente pela Fundação Open Networking. Este protocolo é uma implementação de código aberto que tem sido utilizado pela comunidade acadêmica e por corporações para resolver diversos problemas em suas redes. SDN através de OpenFlow centraliza o controle e padroniza a comunicação entre os comutadores e o controlador da rede, diminuindo a complexidade tanto na borda como no núcleo da rede. Do ponto de vista de QoS, SDN elimina limitações chaves que não permitiram que os mecanismos tradicionais, mencionados anteriormente, fossem implantados até hoje. Algumas das características de SDN que facilitam a implementação de mecanismos de QoS são [ONF 2012]:

- Conhecimento da topologia de rede em tempo real;
- Centralização do algoritmo de roteamento que trabalha sobre a topologia;
- Implantação de forma padronizada das políticas de roteamento sobre a topologia de rede;
- Padronização dos atendimentos para diferentes classes de fluxos no percurso entre ilhas SDNs.

Levando em consideração estas características, o objetivo deste trabalho é elaborar, especificar e desenvolver um modelo analítico-determinístico baseado na teoria de

Network Calculus [Cruz 1991b, Cruz 1991a], que permita implantar e operar, através de um módulo de software, uma rede com capacidade de priorização, marcação, filtragem ou policiamento de fluxos de pacotes a fim de garantir QoS usando o paradigma SDN. Esse modelo avança o estado da arte em SDN já que, pelo nosso melhor conhecimento, não há trabalhos na literatura que proponham modelos como este. Os resultados obtidos nos experimentos, comprovam a viabilidade do NetCalc para subsidiar controle de fluxos garantindo QoS em SDN.

Este trabalho apresenta na seção 2 os principais conceitos relacionados com as SDNs e especifica onde, dentro deste contexto, nosso módulo será elaborado. Logo depois, são mencionados os principais mecanismos de QoS existentes até hoje e como nós reutilizamos algumas características destas arquiteturas. Detalha-se depois o conjunto de definições da teoria de NetCalc utilizadas na elaboração do modelo analítico-determinístico proposto. Mais adiante é feita a modelagem de uma rede, fazendo uso desta teoria, com o fim de, na seção 3, detalhar a proposta do modelo analítico-determinístico que permite priorizar fluxos de pacotes através da teoria de NetCalc em uma rede definida por software. São citados, na seção 4, vários trabalhos relacionados com esta pesquisa, ressaltando a nossa contribuição ao estado da arte. Os experimentos realizados, assim como os resultados obtidos, são apresentados na seção 7. As considerações finais são apresentadas na seção 8.

2. Arcabouço teórico

Esta seção resume os principais tópicos necessários para a compreensão do modelo proposto na seção 3.

2.1. Redes Definidas por Software

As redes definidas por software (SDN) [ONF 2012] são um novo paradigma que visa criar facilmente redes dinâmicas, gerenciáveis e adaptáveis. A arquitetura SDN, (Figura 1), define a rede como uma entidade programável e abstrai os serviços e aplicações de rede das camadas inferiores através de padrões abertos como o protocolo *OpenFlow*.

Na arquitetura *OpenFlow*, o controlador da SDN é responsável pelo reconhecimento dos fluxos de pacotes, determinar uma ou várias ações sobre eles e atualizar as tabelas de fluxos nos comutadores. *OpenFlow* fornece um conjunto de estatísticas que permitem que o controlador conheça a topologia e o estado dos nós da rede (comutador *OpenFlow*) em tempo real, possibilitando o tratamento dinâmico de fluxos.

O módulo implementado neste trabalho pode ser considerado como um serviço de rede acoplado a um controlador. Este módulo usa o modelo analítico proposto para, através de *OpenFlow*, aplicar políticas de priorização e garantir atraso máximo fim-a-fim aos fluxos.

2.2. Qualidade de Serviço

A arquitetura dos **Serviços Integrados** é caraterizada pela reserva de recursos. Antes que a transmissão seja efetuada, uma rota é estabelecida e são reservados recursos ao longo do percurso, através do protocolo RSVP [Braden et al. 1994]. Essa reserva de recursos exige que mudanças sejam realizadas tanto no núcleo quanto na borda da rede. Os **Serviços Integrados** propõem duas classes de serviço além do melhor esforço:

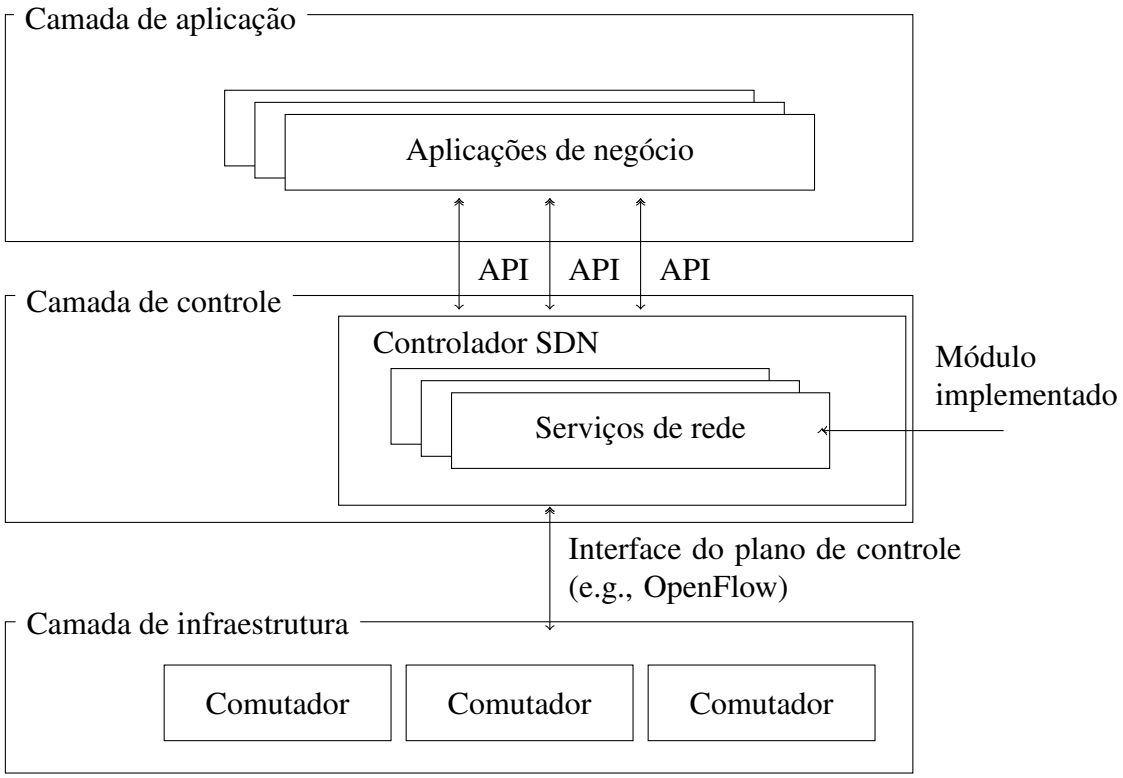


Figura 1. Arquitetura de uma rede definida por software

- Serviço Garantido (*Guaranteed Service*) [Shenker et al. 1997] para aplicações que requerem limites fixos de atraso;
- Serviço de carga controlada (*Controlled-load Service*) [Wroclawski 1997] para aplicações altamente sensíveis a redes congestionadas.

Problemas de escalabilidade, complexidade e custo de implementação são conhecidos e detalhados em [Ni 1999]. Porém, nós reutilizamos o conceito de serviço garantido, a fim de avaliar os fluxos que serão aceitos na rede e fornecer rotas com o menor atraso possível. Este trabalho propõe alocação de recursos não por fluxos individuais e sim por fluxos agregados de uma mesma classificação.

Em **Serviços Diferenciados**, os pacotes são marcados diferentemente visando criar várias classes de fluxos. Os pacotes em classes diferentes recebem serviços diferenciados. Nesta arquitetura a qualidade de serviço é garantida por meio de filas com prioridade [Ni 1999].

Ao contrário dos Serviços Integrados, os Serviços Diferenciados só precisam de mudanças nas bordas da rede. Os roteadores de borda devem classificar os pacotes e os roteadores do núcleo da rede encaminham estes pacotes segundo as prioridades.

Nesta pesquisa a identificação, classificação e a decisão de enfileiramento de fluxos é feita no controlador da rede. Cada fluxo é cadastrado inicialmente no controlador SDN pela fonte transmissora (e.g., instituição fornecedora de fluxos) e deve conter as informações necessárias para a identificação do fluxo bem como os parâmetros de QoS que caracterizam o perfil de tráfego a ser tratado.

2.3. Network Calculus

Network Calculus (NetCalc) é um tópico que fornece um conjunto de resultados matemáticos visando descrever problemas de fluxos encontrados em redes [Cruz 1991b, Cruz 1991a]. NetCalc viabiliza o entendimento de propriedades fundamentais dos serviços integrados, agendamento, alocação de recursos e dimensionamento de atraso, onde, com o fim de fornecer garantias para os fluxos de dados, algumas caracterizações dos tráfegos e serviços da rede são necessários.

Com esse propósito, NetCalc define os conceitos de **curvas de chegadas** e **curvas de serviço**. O subconjunto de definições do arcabouço do NetCalc utilizadas neste trabalho são as seguintes:

- **Notação**
  $R(t)$: função acumulativa de entrada no sistema, estritamente crescente, que determina a quantidade de unidades (bits) em um tempo t
  $R'(t)$: função acumulativa de saída do sistema, estritamente crescente, que determina a quantidade de unidades (bits) atendidos em um tempo t
  $\otimes$: Operador que indica a convolução de duas funções
  $\oslash$: Operador que indica a deconvolução de duas funções
  $\wedge$: Operador que indica o ínfimo de um conjunto de valores

- **Curva de chegada**: Dada uma função crescente α definida para $t \geq 0$ podemos dizer que um fluxo R é restrito por α se e somente se para todo $t \leq s$:

$$R(t) \text{ - } R(s) \leq \alpha(t \text{ - } s), \forall s, t \textit{ onde } 0 \leq s \leq t \tag{1}$$

  Pode-se dizer que R tem α como uma representação abstrata.
- **Curva de chegada remanescente**: Considere um fluxo delimitado por uma curva de chegada α, que atravessa um sistema que oferece uma curva de serviço β, o fluxo de saída é delimitado pela curva de chegada através da operação de deconvolução dada por:

$$\alpha^* = \alpha \oslash \beta \tag{2}$$

- **Curva de serviço**: Considere um sistema S e um fluxo que passa através de S com funções de entrada e saída R e R'. Pode-se dizer que S oferece uma curva de serviço β para o fluxo se e somente se β é estritamente crescente, $\beta(0) = 0$ e:

$$R'(s) - R'(t) \geq \beta(s - t), \forall s, t \textit{ onde } 0 \leq s \leq t \tag{3}$$

  ou

$$R'(t) \geq (R \otimes \beta)(t) \tag{4}$$

- **Atraso**: Menor intervalo de tempo $A(t)$ tal que todas as tarefas iniciadas no instante t sejam completadas para o instante $t + A(t)$ e pode ser especificada da seguinte forma:

$$A(t) = \min\{T : T \geq 0 \wedge R(t) \leq R'(t + T)\} \tag{5}$$

  O atraso no instante t é o atraso que seria experimentado por um *bit* que chega no tempo t se todos os *bits* recebidos antes dele já fossem servidos. Para calcular o limite do atraso o conceito de distância horizontal deve ser utilizado.

- **Distância horizontal**: Para duas funções α e β estritamente crescentes o desvio horizontal $h(\alpha, \beta)$ é dado por:

$$h(\alpha, \beta) = \max_{s \geq 0}\{\min\{T : T \geq 0 \wedge \alpha(s) \leq \beta(s+T)\}\} \tag{6}$$

  A distância horizontal é o valor utilizado para determinar o limite do **atraso** que pode ser descrito como:
- **Limite de atraso**: Para um fluxo restrito por uma curva de chegada α com instantes diferentes e consecutivos de início, sendo servido por uma política de *first-in-first-out* e um sistema S que garante uma curva de serviço β, o atraso para todo t satisfaz:

$$A(t) \leq h(\alpha, \beta), \forall t \geq 0 \tag{7}$$

- **Concatenação**: Considere um fluxo que passa através dos sistemas S_1 e S_2 em sequência. Assuma que S_i oferece uma curva de serviço β_i,
  $i = 1, 2$ ao fluxo. Então a concatenação dos dois sistemas oferece uma curva de serviço $\beta_1 \otimes \beta_2$ ao fluxo.

2.4. Representação do modelo da rede

Neste trabalho a topologia da rede é tratada como um grafo conectado e ponderado onde cada um dos nós oferece uma curva de serviço β. O peso das arestas é dado pelo atraso que experimenta um fluxo caracterizado por uma curva de chegada ao longo das possíveis rotas desde a fonte até o destino, como pode ser observado na Figura 2.

Formalizando matematicamente a rede (Figura 2), para um fluxo que passa através dos nós 1,2,3 temos que:
Seja $G = (V, E)$ uma rede onde V é o subconjunto de nós que compõem a rede, representados pelas suas respetivas curvas de serviço (β),

- V:$\{\beta_1, \beta_2, \beta_3\}$

E é o subconjunto de segmentos de caminhos, formados por enlaces pelos quais o fluxo irá passar da origem X ao destino Y, as quais tem associado um atraso (h), considerando o fluxo de entrada (α) e seu atendimento pelo nó através do serviço (β).

- E:$\{h_{x->y}(\alpha, \beta_1 \otimes \beta_2 \otimes \beta_3)\}$

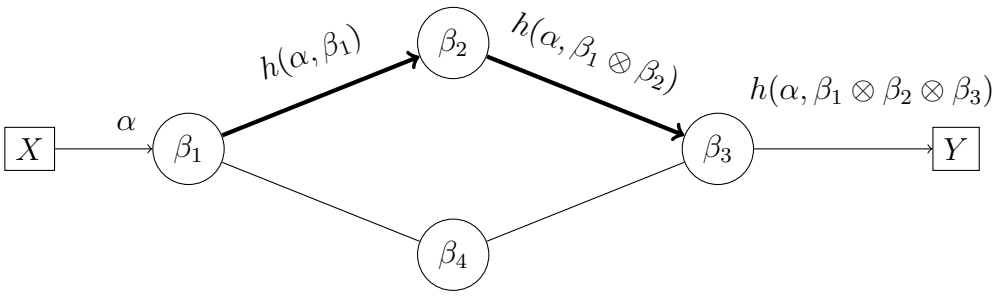


Figura 2. Modelo de rede através de NetCalc

Sendo assim, ao escolher um caminho P qualquer, desde uma fonte transmissora até o destino que requisita a transmissão, o atraso máximo A_{max}, experimentado pelo

fluxo α através de P será:

$$A_{\max} \leq h(\alpha, (\beta_i \otimes \beta_{i+1} ..., \otimes \beta_m)) \quad (8)$$

Observe que no grafo não é considerado nem o enlace inicial de X para o primeiro nó, nem o atraso associado. A explicação desta consideração é que o controlador da SDN somente possui informação sobre os comutadores SDN e não dos equipamentos (e.g., *hosts*) conectados em cada um deles.

3. Modelo para priorização de tráfego com Network Calculus

O grau de sensibilidade dos fluxos priorizados às métricas de QoS como o atraso, a variação de atraso, a largura de banda e a perda de pacotes, vai depender do tipo de fluxo. Para vídeos interativos (vídeo conferência) [Hattingh, Christina and Szigeti 2004] recomenda uma perda de pacotes menor que 1%, atraso menor que 150 ms e uma variação de atraso não maior que 30 ms. Por sua vez, para *streaming* de vídeo, o mesmo autor sugere uma perda menor que 5% e um atraso não maior que 5 segundos. Para este caso não há valores significativos para a variação de atraso.

Visando fornecer atendimento diferenciado para distintas classes de fluxos, nós precisamos que os nós da rede ofereçam mais de uma curva de serviço. Sem perda de generalidade, neste modelo é proposta a criação de três classes de fluxos: **C1, C2, C3** sendo C1 a classe com maior prioridade e C3 com a menor prioridade. Cada classe tem associada a sua respectiva fila de atendimento as quais, neste trabalho, são do tipo *first-in-first-out* (FIFO), como é observado na Figura 3.

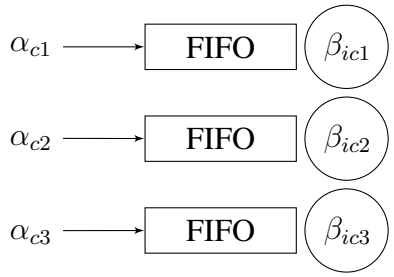


Figura 3. Atendimento diferenciado nos canais de saída dos nós da rede

Uma nova rede pode ser definida como:
Seja $G = (V, E)$ uma rede onde V é o subconjunto de nós que compõem a rede, representados pelas suas respetivas curvas de serviço específicas para cada classe de fluxo ($\beta_{c1,c2,c3}$). E é o subconjunto de segmentos de caminhos, formados por enlaces pelos quais o fluxo irá passar da origem X ao destino Y, os quais tem associado um atraso (h), considerando o fluxo de entrada (α) e seu atendimento **diferenciado** pelo nó através do serviço específico requisitado ($\beta_{c1|c2|c3}$). Ver (Figura 4) :

- V:$\{(\beta_{1c1}, \beta_{1c2}, \beta_{1c3}), (\beta_{2c1}, \beta_{2c2}, \beta_{2c3}), (\beta_{3c1}, \beta_{3c2}, \beta_{3c3})\}$;
- E:$\{h_{X->Y}(\alpha_{cm}, \beta_{1cm} \otimes \beta_{2cm} \otimes \beta_{3cm})\}$

Novamente escolhendo um caminho P, desde uma fonte transmissora até o destino que requisita a transmissão, o atraso máximo $A_{\max}$, experimentado pelo fluxo α com atendimento diferenciado Cm através de P será:

$$A_{\max} \leq h(\alpha_{cm}, (\beta_{icm} \otimes \beta_{(i+1)cm} ..., \otimes \beta_{jcm})), \text{ } \textit{onde } i, j \in \mathbb{N}, j = |P| \quad (9)$$

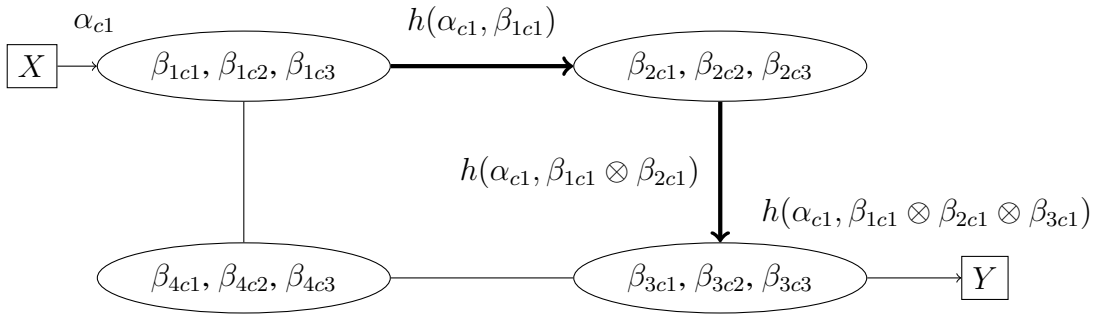


Figura 4. Modelo de rede com atendimento diferenciado

3.1. Agregação de fluxos

Até o momento nosso modelo permite atender três fluxos, se e somente se, eles precisarem de classes de serviços diferentes (i.e., um fluxo por classe). Com o fim de agregar fluxos da mesma classe de atendimento, nosso modelo deve possuir uma forma de somar os fluxos admitidos no sistema, visando dar como resultado o atraso máximo experimentado, desta vez não por um fluxo, e sim pelos fluxos agregados de uma mesma classe.

Para um conjunto de fluxos da forma α que passam pelo caminho P, o atraso máximo experimentado pelo fluxo resultante da agregação é dado pela expressão:

$$A_{\max} \leq h(\alpha_{1cm} + \alpha_{2cm} + \alpha_{3cm}..., +\alpha_{ncm}, (\beta_{icm} \otimes \beta_{(i+1)cm}..., \otimes\beta_{jcm})) \qquad (10)$$

Em teoria n pode chegar até o infinito, na prática só serão aceitos no sistema um conjunto de fluxos, onde $A_{\max}$ experimentado pela agregação, seja menor ou igual ao menor dos atrasos especificados por cada fluxo cadastrado no **controlador** da rede.

3.2. Melhor caminho fim-a-fim

Múltiplos algoritmos para encontrar o caminho mais curto entre os vértices de um grafo foram propostos ao longo do tempo. Um dos mais utilizados nas redes (e.g., IS-IS [Smit and Li 2004], OSPF [J.Moy 1998]) é o algoritmo de Dijkstra [Dijkstra 1959].

Qualquer algoritmo, que para encontrar o caminho mais curto leve em consideração o valor associado a cada aresta, pode utilizar nosso modelo analítico. Neste trabalho utilizaremos Dijkstra, sendo este relativamente simples e eficiente (i.e., $O(n \log(n))$), onde n é a quantidade de vértices do grafo.

Um aspecto a ser considerado na computação do melhor caminho fim-a-fim é que, segundo foi demostrado na teoria de *Network Calculus*, a soma dos atrasos experimentados em cada nó da rede (i.e., $h(\alpha, \beta_1) + h(\alpha^*, \beta_2) + h(\alpha^*, \beta_3)... + h(\alpha^*, \beta_n)$), é maior do que a convolução das curvas de serviço oferecidas pelos nós no caminho (i.e., $h(\alpha, \beta_1 \otimes \beta_2 \otimes \beta_3... \otimes \beta_n)$). Portanto devemos adaptar o algoritmo de Dijkstra visando ser coerente com as inequações (8)(9) e (10), como é mostrado no Algoritmo 1. Considere que quando se acessa a curva de serviço β, em um comutador, ela corresponde ao atendimento diferenciado solicitado pelo fluxo, caraterizado por α.

Algoritmo 1: Dijkstra utilizando NetCalc

```
Data: rede, fonte, destino, alpha
Result: melhor_rota, atraso_máximo
begin
    atraso[fonte] := 0 ;                                  // Atraso para chegar em [comutador]
    alpha_agregada := fonte.alpha + alpha ;                                  // Agregação, (10)
    atraso_máximo := 0;
    for comutador ∈ rede do
        if comutador ≠ fonte then
            atraso[comutador] := INFINITO;
            anterior[comutador] := INDEFINIDO;
        end
        adiciona comutador em nao_visitados
    end
    while nao_visitados ≠ ∅ do
        /* c é a fonte na 1ra iteração                                                    */
        c := comutador ∈ nao_visitados com menor atraso[c];
        remover c de nao_visitados;
        for vizinho v de c do
            /* Atraso para chegar em v, (8)(9)                                          */
            a := h(alpha_agregada, c.beta_{c->v});
            if a < atraso[v] then
                /* v.betas, curvas de serviço de todas as portas do comutador
                   */
                v.betas = c.beta_{c->v} ⊗ v.betas ;                          // Concatenação
                atraso[v] := a ;                                  // Atualização do atraso
                anterior[v] := c ;                                  // Construção da rota
            end
            if v = destino then
                atraso_máximo := h(alpha_agregada, convolução(anterior[].betas));
                /* Rota fonte->destino                                                  */
                return anterior[].invertido(), atraso_máximo ;
            end
        end
    end
end
```

4. Trabalhos relacionados

Vários trabalhos apresentam o uso do *OpenFlow* para a otimização da transmissão de fluxos, onde o melhor esforço da Internet não atende os requisitos mínimos em termos de qualidade de serviço. Uma destas propostas implementa um módulo que é denominado OpenQoS integrado ao controlador *OpenFlow*, que utiliza um mecanismo de diferenciação de tráfego. Este módulo permite tratar dinamicamente, de forma diferenciada, os fluxos de vídeo escolhendo a rota que mais se adequa aos parâmetros de QoS requisitados [Egilmez and Dane 2012].

Em outro trabalho Egilmez propõe a utilização de codificação de vídeo escalável criando com isso diferentes camadas de qualidade de serviço e priorização, que serão usadas de acordo com as condições da rede [Egilmez et al. 2011]. A priorização é feita estaticamente no controlador no instante de associar uma rota a um fluxo e não existe uma alocação de recursos no *switch*.

Outro trabalho [Owens II and Durresi 2013] propõe ainda a implantação de um módulo no controlador *OpenFlow* que manipula métricas de QoS. O protocolo proposto visa garantir o entendimento mútuo entre o transmissor e o receptor na hora da solicitação de um vídeo e a procura de uma rota com qualidade de serviço. Basicamente é uma troca de mensagens que resulta no estabelecimento ou não de uma sessão. O autor propõe modificações ao protocolo *OpenFlow* para implementar a sua solução, acrescentado métricas de Qualidade de Serviço nas descrições das filas dos comutadores *OpenFlow*.

Em um trabalho mais recente Qin [Qin et al. 2014] utiliza NetCalc e Algoritmo Genético para definir dinamicamente diferentes oportunidades e agendamento de atendi-

mento de fluxos por uma estrutura SDN para cenários de Internet das Coisas (IoT – Internet of Things) em que são consideradas redes sem fio heterogêneas. O sistema implementado (uma extensão do *Multinetwork INformation Architecture* (MINA) ao ser integrado a veículos elétricos, demonstrou ganhos de eficiência na capacidade de atendimento em transmissões de dados gerados em cenário de IoT baseada em redes heterogêneas.

A contribuição de nosso trabalho é explorar o NetCalc como uma ferramenta para que o controlador SDN possa fazer a seleção de rotas para o atendimento a diferentes classes de fluxos, levando em consideração requisitos de QoS.

5. Experimentos e resultados

Visando validar os aspectos teóricos deste trabalho, descritos na seção 3, foi elaborado um ambiente virtual composto pelo controlador de rede Floodlight [Shah et al. 2013] e uma topologia de comutadores e estações (i.e., $ht1, ht2, ht3, ht4$), criada através da ferramenta Mininet [Lantz et al. 2010], como pode ser observado na Figura 5. Este ambiente permite a simulação de uma rede definida por software que opera através do protocolo OpenFlow.

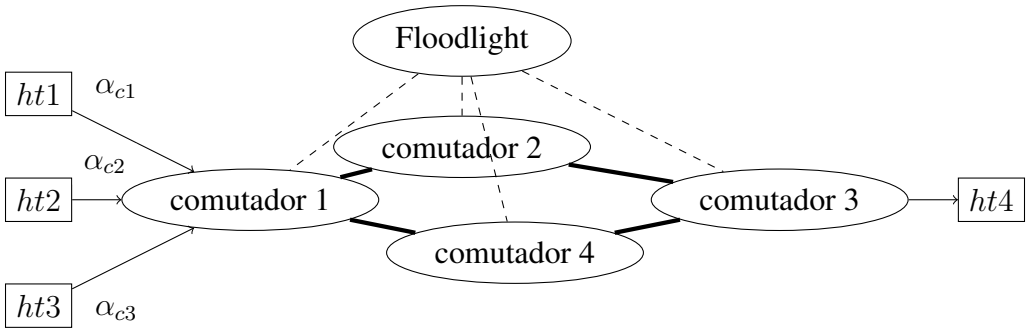


Figura 5. Rede virtual SDN.

5.1. Módulo para o controlador SDN

Foi desenvolvido um **módulo** dentro do controlador da rede que permite cadastrar as especificações dos fluxos de dados a serem transmitidos. Os parâmetros a seguir foram utilizados para a caracterização do tráfego:

- **ipsrc**: Endereço da fonte transmissora;
- **psrc**: Porta utilizada na transmissão;
- **ac**: Função envelope que carateriza o tráfego;
- **class**: Classe de fluxo que especifica o tipo de atendimento (C1, C2, C3);
- **delay**: Atraso máximo fim-a-fim suportado.

Uma vez que chega um pacote de rede ao controlador, este módulo compara os parâmetros cadastrados com os atributos do pacote (i.e., ipsrc, psrc). No caso em que o pacote seja identificado positivamente e o atraso virtual máximo da rota fim-a-fim, calculado através do algoritmo 1, seja menor ou igual ao atraso máximo cadastrado, o fluxo é aceito na rede. Logo depois é definida a rota com a priorização requisitada nas tabelas de fluxos nos comutadores.

O módulo é responsável pela distribuição dos recursos disponíveis nas portas de saída dos comutadores *Openvswitch* [Pfaff et al. 2009]. Esta distribuição é correspondente ao grau de priorização dos fluxos, e é realizada utilizando filas *Hierarchical Token*

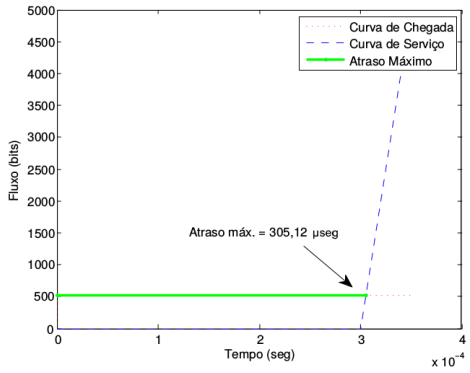
Bucket [Hubert and Others 2002], através do protocolo *The Open vSwitch Database Management Protocol* [Pfaff and Davie 2013]. Isto permite que fluxos com classes diferentes experimentem atrasos de acordo com a utilização de sua fila de atendimento.

Visando obter os resultados fornecidos pela teoria de NetCalc o módulo implementado utiliza a biblioteca DiscoDNC[1]. Os principais motivos para a escolha desta biblioteca, depois da análise de várias [Boyer 2010], foram a linguagem de programação em que é escrita (i.e., Java), que é a mesma do controlador Floodlight, e o fato do DiscoDNC fornecer classes prontas que representam as definições apresentadas na seção 2.3.

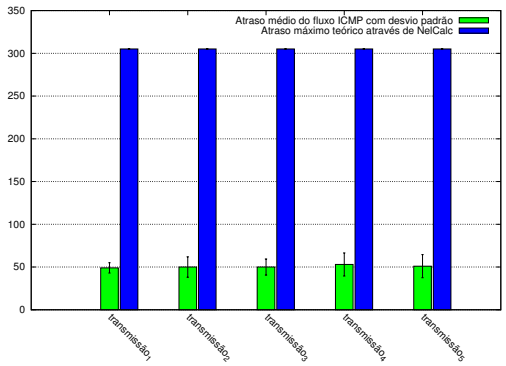
5.2. Descrição do experimento

O primeiro objetivo do experimento foi validar o correto funcionamento do ambiente simulado. Para isso foi modelado um fluxo com origem-destino $ht1 - ht4$ e prioridade alta *C1*. A curva de chegada deste fluxo foi considerada como $\alpha = \gamma_{r,b} = \gamma_{64,64}$, isto é uma taxa de 64 bytes com rajadas também de 64 bytes. Inicialmente todas as portas dos comutadores da rede possuem uma capacidade de $100Mbps$.

Várias análises têm sido realizadas com relação ao desempenho dos comutadores *Openvswitch* em termos de atraso de atendimento [Jarschel et al. 2011]. Esta métrica varia entre 35 e 100 microssegundos e vai depender fundamentalmente da dimensão da tabela de fluxos e do tamanho dos pacotes a serem processados. Neste experimento foi adotado um atraso de $T = 0.0001s$, sendo que as curvas de serviço das filas de saída dos comutadores estão dadas por $\beta 1 = \beta 2 = \beta 3 = \beta 4 = \beta_{R,T} = \beta_{12500000,0.0001}$. A rota fim-a-fim foi inserida manualmente através dos comutadores 1,2 e 3. Aplicando a definição (8) o atraso máximo teórico é de 305.12 microssegundos como pode ser observado na Figura 6(a).



(a) Atraso máximo entre $\gamma_{64,64}$ e $\beta_{12500000,0.0003}$



(b) Atrasos dos fluxos ICMP e atraso máximo teórico com NetCalc

Figura 6. Comparação entre os atrasos teóricos através de NetCalc e reais no ambiente virtual.

Visando realizar uma comparação entre os atrasos que o fluxo modelado experimentou teoricamente e um fluxo real com as mesmas características, foram feitas 5 transmissões ICMP de 60 s desde $ht1$ até $ht4$. O resultado desta comparação está refletida na Figura 6(b), onde o atraso máximo do fluxo ICMP foi de 66.5 microssegundos. Então, aplicando (8) novamente temos que: $A_{\max} \leq \frac{b}{\min(R1,R2,R3)} + \sum_{n=1}^{3} T_n$

[1]The Disco Deterministic NetCalc, http://disco.cs.uni-kl.de/index.php/projects/disco-dnc

Tabela 1. Fluxos cadastrados no controlador SDN

	ipsrc	psrc	class
Fluxo 1	ht1	5001	C1
Fluxo 2	ht2	5002	C2
Fluxo 3	ht3	5003	C3

Logo, $66.5 \leq 5.12 + 3T$, então, $T \geq 20.46$ microssegundos. Este resultado significa que o atraso de atendimento T, dos comutadores de nosso experimento está 14.54 microssegundos abaixo do limite inferior da faixa de 35 a 100 microssegundos obtida em [Jarschel et al. 2011]. Tal comportamento pode ser justificado por diversos fatores, dentre eles, as melhorias com relação ao desempenho na implementação do *Openvswitch*, desde a data em que foi feito o trabalho referenciado até hoje.

O principal objetivo da elaboração deste experimento é aplicar o modelo analítico-determinístico proposto na seção 3, sobre a rede virtual SDN elaborada. Para isso faremos uso do módulo desenvolvido no controlador SDN, explicado anteriormente. Inicialmente foi diminuída a capacidade dos enlaces do comutador 4 para $10Mbits/seg$. Esta modificação permite comprovar se o Algoritmo 1 escolhe a melhor rota fim-a-fim. As capacidades dos enlaces foram distribuídas de acordo ao grau de priorização, sendo que, $C1 = 50\%, C2 = 30\%, C3 = 20\%$. Foram cadastrados no controlador 3 fluxos com perfis de tráfego diferentes como se mostra na Tabela 1. Logo depois, 3 transmissões simultâneas TCP de 100MBytes foram feitas com destino $h4$.

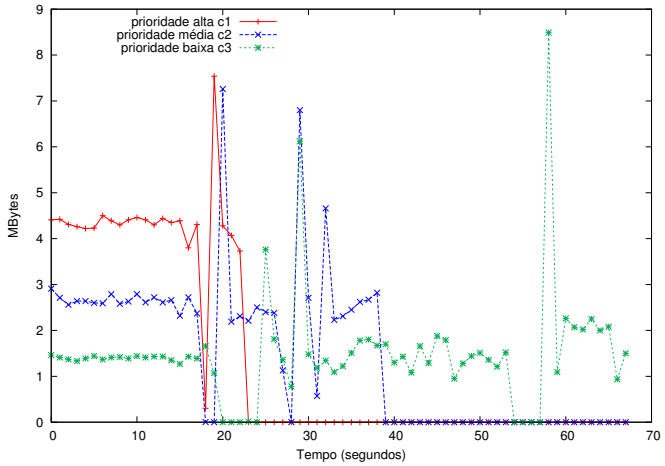


Figura 7. Transmissão simultânea de 3 fluxos com diferentes graus de priorização

No resultado destas transmissões, observou-se na Figura 7, como são atendidos os fluxos de acordo com o grau de priorização de cada um. O fluxo com prioridade alta manteve uma taxa média aproximada de 34 Mbps e finalizou em 23 s. Por sua vez o fluxo com prioridade média foi atendido de forma a manter uma taxa média de transmissão de 20.4 Mbps. Este fluxo transferiu os 100MBytes em 39 s.

Por último o fluxo de menor prioridade finalizou a transmissão em 68.4 s com uma taxa média aproximada de 11.68 Mbps. Evidentemente o Algoritmo 1, teve sucesso na procura da rota com melhor QoS em termos de atraso fim-a-fim e foi garantida a

priorização dos fluxos.

6. Considerações finais

Neste trabalho foi criado um modelo analítico-determinístico baseado na teoria de *Network Calculus*, permitindo implementar mecanismos de QoS em uma rede definida por software. Visando validar este modelo, foi desenvolvido um módulo acoplado a um controlador de rede que mostrou a viabilidade de utilização deste tipo de abordagem em um ambiente onde é feito um controle centralizado da rede.

O modelo elaborado faz uso das vantagens de programabilidade da rede que oferecem as SDNs. Considera-se que os resultados obtidos nesta pesquisa podem contribuir no aprimoramento dos mecanismos de QoS existentes até hoje, assim como as teorias desenvolvidas para a engenharia de tráfego. O gerenciamento de redes pode ser beneficiado também com o modelo analítico elaborado, pois este não precisa do monitoramento contínuo sobre a infraestrutura de rede para obter métricas de QoS.

Em estudos futuros pretende-se ampliar o modelo proposto de forma a incorporar outras métricas de QoS. Além disso será realizado um aprofundamento na teoria de Netcal, visando refletir modelos de redes mais complexos, onde sejam levados em consideração aspectos fundamentais como a capacidade das filas nas portas dos comutadores, assim como variabilidade do tamanho dos pacotes de um fluxo com igual grau de priorização.

Referências

Blake, S., Microsystems, S., and Wang, Z. (1998). An Architecture for Differentiated Services. *RFC*, pages 1–37.

Boyer, M. (2010). NC-maude: A rewriting tool to play with network calculus. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6415 LNCS, pages 137–151.

Braden, R., Clark, D., and Shenker, S. . (1994). Integrated Services in the Internet Architecture: an Overview. *IETF RFC 1633, July*, pages 1–28.

Cisco (2014). The Zettabyte Era: Trends and Analysis, White Paper. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html`. Acessado em 31 de Março de 2015.

Cruz, R. (1991a). A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131.

Cruz, R. (1991b). A calculus for network delay. II. Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271.

Egilmez, H. and Dane, S. (2012). OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. *Signal & Information ...*, pages 1–8.

Egilmez, H. E., Gorkemli, B., Tekalp, A. M., and Civanlar, S. (2011). Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing. *2011 18th IEEE International Conference on Image Processing*, pages 2241–2244.

Hattingh, Christina and Szigeti, T. (2004). End-to-end QoS network design: quality of service in LANs, WANs, and VPNs.

Hubert, B. and Others (2002). Linux advanced routing & traffic control HOWTO.

Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., and Tran-Gia, P. (2011). Modeling and performance evaluation of an OpenFlow architecture. *2011 23rd International Teletraffic Congress (ITC)*, pages 1–7.

J.Moy (1998). OSPF Version 2. *IETF RFC 2328, April*, pages 1–245.

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. M., Peterson, L. L., Rexford, J., Shenker, S., Turner, J. S., and Louis, S. (2008). OpenFlow: enabling innovation in campus networks. *Computer Communication Review*, 38:69–74.

Ni, L. (1999). Internet QoS: a big picture. *IEEE Network*, 13(2):8–18.

ONF (2012). Software-Defined Networking : The New Norm for Networks. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf`. Acessado em 31 de Março de 2015.

Owens II, H. and Durresi, A. (2013). Video over Software-Defined Networking (VSDN). In *2013 16th International Conference on Network-Based Information Systems*, pages 44–51. IEEE.

Pfaff, B. and Davie, B. (2013). The Open vSwitch Database Management Protocol. Technical report, RFC 7047, IETF, December.

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M., and Shenker, S. (2009). Extending Networking into the Virtualization Layer. In *8th ACM Workshop on Hot Topics inNetworks*, volume VIII, page 6.

Qin, Z., Denker, G., Giannelli, C., Bellavista, P., and Venkatasubramanian, N. (2014). A Software Defined Networking architecture for the Internet-of-Things. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9. IEEE.

Rosen, E., Viswanathan, A., and Callon, R. (2001). Multiprotocol label switching architecture. *Vasa*, pages 1–62.

Shah, S. A., Faiz, J., Farooq, M., Shafi, A., and Mehdi, S. A. (2013). An architectural evaluation of SDN controllers. In *2013 IEEE International Conference on Communications (ICC)*, pages 3504–3508. IEEE.

Shenker, S., Partridge, C., and Guerin, R. (1997). RFC-2212: Specification of Guaranteed Quality of Service.

Smit, H. and Li, T. (2004). Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE). *RFC 3784*.

Wroclawski, J. (1997). RCF-2211: Specification of the Controlled-Load Network Element Service.

Modelo e Solução para o Problema de Posicionamento de Agregadores em Redes Elétricas Inteligentes

Guilherme Rolim, Célio Vinicius Neves de Albuquerque, Igor Monteiro Moraes

[1]Laboratório MídiaCom, PGC-TCC
Instituto de Computação - Universidade Federal Fluminense
Niterói, Rio de Janeiro, Brasil

grolim@id.uff.br, {celio,igor}@ic.uff.br

Abstract. *In a Smart Grid infrastructure, smart meters are responsible for keeping track of user energy consumption. This data is periodically sent to one or more data aggregation points (DAP), typically via wireless communication. Efficiently choosing the best positions for installing DAPs is an NP-Complete problem and therefore a difficult task, specially in big cities that may contain thousands of smart meters in a single neighbourhood. Nevertheless, network planning has a major impact on network performance. This work proposes an adaptation of the DAPs positioning problem to a problem in the optimization area known as the Set Covering Problem and a heuristic to solve it. Heuristic and linear programming techniques are compared and results show that the heuristic is capable of obtaining solutions 3% close to the optimal while accelerating 3.5 times the execution duration and reducing 10.5 times the memory consumption in comparison to the exact method.*

Resumo. *Em uma rede elétrica inteligente (Smart Grid) , medidores inteligentes armazenam dados de consumo energético de cada residência que são enviados periodicamente a um ou mais agregadores, tipicamente via comunicação sem fio. Escolher eficientemente as melhores posições para agregadores é um problema NP-Completo e, portanto, torna-se uma tarefa difícil, principalmente em grandes cidades que podem conter milhares de medidores em um único bairro. Este trabalho propõe uma adaptação do problema de posicionamento de agregadores em Smart Grids para um problema conhecido da área de otimização como Set Covering Problem e uma heurística para resolvê-lo. As soluções da heurística são comparadas às respostas exatas e avaliadas quanto ao seu desempenho. Os resultados mostram que a heurística é capaz de obter resultados 3% próximos ao ótimo acelerando 3,5 vezes o tempo de execução e reduzindo em 10,5 vezes o consumo de memória se comparado ao método exato.*

1. Introdução

As redes elétricas inteligentes (*Smart Grids*) são uma evolução das redes elétricas atuais, pois permitem a comunicação de dados entre consumidores e produtores, além da distribuição de energia elétrica. Esta comunicação, ausente nas redes elétricas convencionais, permite que as empresas elétricas utilizem as informações geradas por consumidores durante a geração, transmissão e distribuição de energia. No entanto, para que as redes

elétricas inteligentes funcionem é necessária a instalação de agregadores de dados e medidores inteligentes, isto é, dispositivos eletrônicos dotados de interfaces de rede para que possam se comunicar. Os medidores e agregadores fazem parte da infraestrutura de medição avançada (*Advanced Metering Infrastructure* - AMI) [Wenpeng 2009], que é uma das principais propostas de aplicação para *smart grids* e cujo objetivo é prover comunicação entre consumidores e concessionária. Nessa infraestrutura, cada residência contém um medidor inteligente responsável por armazenar informações periódicas do consumo de energia elétrica dessa residência. Os dados coletados por estes medidores são transmitidos a um agregador de dados, tipicamente via meio sem fio, evitando gastos na criação de infraestrutura para redes cabeadas. Os agregadores, por sua vez, podem transmitir os dados coletados de uma vizinhança à concessionária de energia. Forma-se, assim, uma rede de comunicação de curta e média distância. Não há uma padronização sobre qual tecnologia deve ser utilizada para comunicação medidor-agregador e agregador-concessionária. Para a primeira, podem ser usados padrões de comunicação para curta distância, como o IEEE 802.15.4 e o IEEE 802.11. Para a segunda, podem ser usadas tecnologias com maior alcance como a rede celular GPRS, 3G, 4G ou o padrão IEEE 802.16.

Um dos maiores desafios para o desenvolvimento de uma infraestrutura de medição avançada é planejar as posições de agregadores de dados. Como em uma rede elétrica inteligente cada residência possui um medidor, verificar manualmente as posições mais adequadas para que agregadores os cubram é uma tarefa praticamente inviável, principalmente em vizinhanças com alta densidade demográfica. Além disso, deve-se levar em consideração a capacidade máxima de cada agregador e posicioná-los de forma que não sobrecarreguem. O planejamento manual para uma cidade inteira, por exemplo, envolveria analisar milhares de medidores, um para cada residência. Adicionalmente, o mal planejamento da AMI resulta em baixo desempenho. Okabayashi observou que uma AMI mal planejada resulta em alto atraso, alto índice de perda de pacotes e baixa resiliência no caso de falha de agregadores [Okabayashi 2014].

Este trabalho propõe uma nova metodologia para o posicionamento automático de agregadores de dados em uma rede elétrica inteligente. As posições geográficas de medidores, as possíveis posições de instalação de agregadores e o alcance médio de um enlace são adaptados para um problema de otimização NP-completo conhecido na literatura, o *Set Covering Problem* (SCP) [Karp 1972]. Solucionando o SCP é possível obter a menor quantidade de agregadores possível para que todos os medidores sejam cobertos e as posições em que deverão ser instalados. Uma heurística é proposta para solucionar o SCP através da divisão do problema em subconjuntos que são resolvidos separadamente de forma exata. A união das soluções de cada subconjunto compõe a solução do problema como um todo. Resultados mostram que essa heurística é capaz de reduzir a quantidade de memória e o tempo de execução sacrificando pouco a qualidade da solução.

O restante deste trabalho está organizado como segue. A Seção 2 explica o conceito de uma infraestrutura de medição avançada. A Seção 3 apresenta trabalhos relacionados. A Seção 4 descreve a formulação matemática do *Set Covering Problem* e como o problema de planejamento com e sem múltiplos saltos foi adaptado a ele. A Seção 5 explica a metodologia proposta para solucionar o problema adaptado. A Seção 6 avalia o desempenho da heurística proposta e a vantagem de se utilizar múltiplos saltos. Por fim, a Seção 7 conclui esse trabalho e apresenta trabalhos futuros.

2. A Infraestrutura de Medição Avançada

Em uma infraestrutura de medição avançada (AMI) [Wenpeng 2009] agregadores inteligentes atuam como nós intermediários na comunicação, conectando medidores até uma central. A AMI permite que consumidores tenham informação sobre a rede elétrica, podendo verificar a tarifação e dados de consumo energético em tempo real. Por outro lado, essa infraestrutura também permite às concessionárias obter informações detalhadas sobre o consumo energético de seus usuários, garantindo maior controle na distribuição de energia e evitando desperdícios.

O principal desafio da AMI é conectar os dispositivos inteligentes de forma a se evitar custos desnecessários com infraestrutura [Guimarães et al. 2013]. Por este motivo, é indicado que a comunicação entre dispositivos da AMI seja realizada via meio sem fio ou através de infraestruturas já existentes, como a da própria rede elétrica ou de telefonia. A Figura 1 exibe um exemplo de AMI. Os medidores de cada casa, representados por círculos na figura, conectam-se a um agregador, que transmite dados gerados por esses medidores à concessionária utilizando uma tecnologia de longo alcance. Os agregadores, representados por triângulos, podem ser instalados em postes, como mostra a figura.

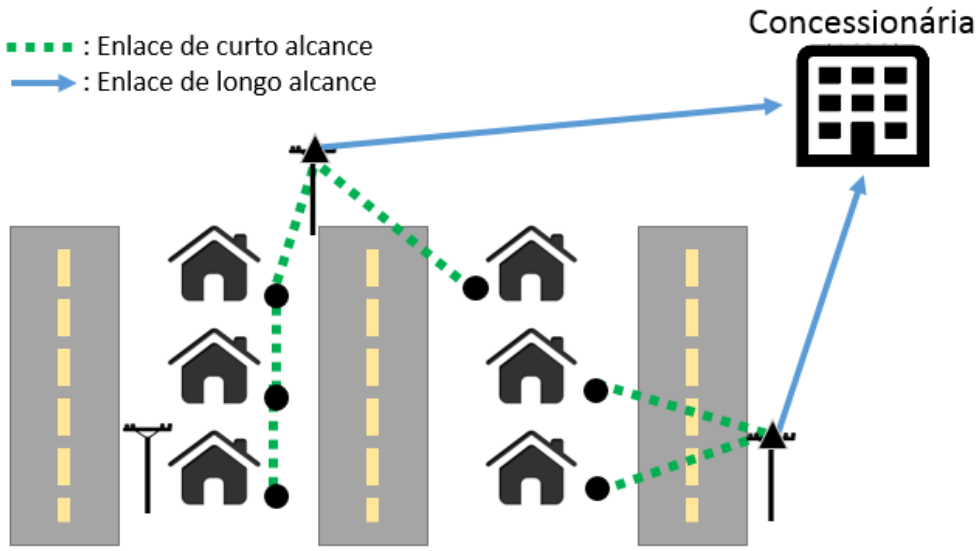


Figura 1. Exemplo de infraestrutura de medição avançada.

Este trabalho assume que a comunicação entre medidores e agregadores é realizada exclusivamente via meio sem fio. Não é escopo deste trabalho abordar a tecnologia utilizada entre agregadores e concessionária. Considera-se apenas que todos os agregadores são capazes de encaminhar os dados gerados pelos medidores.

3. Trabalhos Relacionados

Souza *et al.* propõem um mecanismo de posicionamento de agregadores para uma rede com RF-Mesh [Souza et al. 2013]. Nesse trabalho os autores introduzem um algoritmo que identifica a melhor posição para instalação de agregadores baseando-se no número de saltos obtidos pelos métodos *Breadth-First Search*, algoritmo de Dijkstra e algoritmo de Bellman-Ford. A identificação da melhor posição é realizada através

de uma busca exaustiva que compara o número de saltos para cada posição analisada. Adicionalmente, o algoritmo permite o posicionamento de mais de um agregador. Neste caso, o método de clusterização *K-means* é executado previamente, dividindo os medidores em clusteres. Para cada cluster, a busca exaustiva é realizada nas posições mais próximas de seu centro. Os autores desse trabalho também propõem uma formulação matemática para o problema e conseguem resolvê-lo através de Programação Linear Binária, tendo como entrada a quantidade de agregadores que se deseja utilizar [Souza, Gustavo B. de C. et al. 2013].

O trabalho de Leon [Leon 2011] desenvolve um guia de boas práticas para o planejamento de uma infraestrutura de medição avançada que utiliza redes em malha e dispositivos que operam na frequência de 902-928 MHz. O autor utiliza a ferramenta EDX SignalPro [EDX Wireless 2014] para validar suas propostas em uma amostra de vizinhança que contém 40 mil medidores. Contudo, o autor não teve como objetivo obter o melhor posicionamento de agregadores.

A grande desvantagem dos trabalhos citados anteriormente está na necessidade de se indicar previamente o número de agregadores que devem ser posicionados. Descobrir a menor quantidade de agregadores necessária e suas posições não é uma tarefa trivial. O método proposto neste trabalho difere-se dos citados acima pois obtém a menor quantidade necessária de agregadores além de suas respectivas posições. Para isso, o problema de planejamento de uma AMI é adaptado ao *Set Covering Problem*. O problema também é adaptável a problemas similares conhecidos como *Facility Location Problems* [Farahani et al. 2012].

Os autores deste artigo, em um trabalho anterior, apresentaram uma ferramenta para planejamento de *smart grids* [Rolim et al. 2014]. Descrevem-se as funcionalidades e a interface gráfica dessa ferramenta. No entanto, o problema da alocação automática de agregadores é tratado como uma caixa-preta. Um vídeo de demonstração desta ferramenta foi vencedor da competição de estudantes da SmartGridComm 2014 [Rolim 2014].

4. Adaptação do *Set Covering Problem*

Esta seção detalha o *Set Covering Problem* e discute como o planejamento automático de agregadores em *smart grids* pode ser adaptado a este problema.

4.1. Formulação Matemática do SCP

O SCP clássico é descrito da seguinte forma: dado um conjunto M de tamanho m e n subconjuntos $S_j \subseteq M$, onde $j \in N = 1, ..., n$ cada um contendo um custo não negativo c_j, o objetivo é minimizar o custo de subconjuntos S_j de tal forma que cada elemento de M pertença a pelo menos um destes subconjuntos. Uma formulação matemática deste problema comum na literatura é descrita como:

$$min \sum_{j=1}^{n} c_j.x_j \tag{1}$$

$$\text{Sujeito a: } \sum_{j \in N} a_{ij} x_j \geq 1, i \in M \tag{2}$$

$$x_j \in 0, 1, j \in N. \tag{3}$$

A variável x_j é igual a 1 se o subconjunto S_j pertence à solução e 0 caso contrário. O coeficiente a_{ij} é 1 se o elemento i pertence a S_j e 0 caso contrário. A matriz $A = (a_{ij})$, $i = 1, ..., m$, $j = 1, ..., n$ é chamada de matriz de cobertura. Na matriz de cobertura cada linha representa um elemento a ser coberto e cada coluna um subconjunto. A Restrição (1) garante que todos os elementos do conjunto M serão cobertos por pelo menos um subconjunto. A função de minimização consta com a variável c_j que indica o custo de escolha de um determinado subconjunto. Este trabalho aborda apenas a variante do SCP de custo único, introduzida por [Toregas et al. 1971]. Por este motivo, a variável c_j pode ser ignorada, visto que, o custo de escolha de cada subconjunto é idêntico e não influencia na função de minimização.

A Figura 2 ilustra uma matriz de cobertura do SCP. De forma simples, o objetivo é selecionar o menor número de colunas possível de forma que todas as linhas sejam cobertas. Uma linha L é coberta por uma coluna C se o par (L, C) for representado pelo valor 1. Note que para este exemplo, as Colunas 0, 1 e 2 cobrem todas as linhas do problema, logo constituem uma solução válida. No entanto, o objetivo do SCP é minimizar o número de conjuntos na solução. As Colunas 0 e 3 compõem o menor número de colunas possível para que todas as linhas sejam cobertas constituindo a solução ótima para o exemplo apresentado.

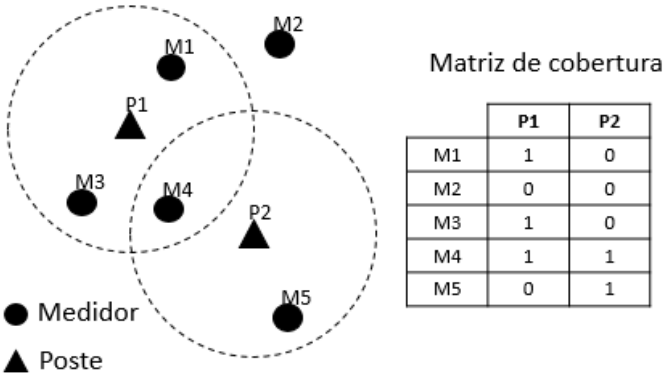
	0	1	2	3
0	1	0	0	0
1	0	1	0	1
2	0	0	1	1

Figura 2. Uma matriz de cobertura do *Set Covering Problem*.

4.2. O Problema de Planejamento de Agregadores como um SCP

O objetivo do SCP é determinar a menor quantidade de subconjuntos que contenham todos os elementos que devem ser cobertos. Para o planejamento de uma *smart grid*, os elementos a serem cobertos são os medidores e os subconjuntos são formados pelos medidores que serão cobertos se um agregador for adicionado à determinada posição geográfica. Para facilitar o entendimento, essas posições são consideradas como postes elétricos. Como no exemplo da Figura 2, os medidores podem ser identificados pelas linhas e os postes pelas colunas. Ao se determinar a menor quantidade de colunas necessárias, é possível saber em quais postes os agregadores devem ser instalados para que toda a rede seja coberta.

Para que a matriz de cobertura possa ser criada, é preciso obter as posições geográficas dos medidores e dos postes. Através destas posições e do alcance dos agregadores, é possível determinar quais agregadores alcançam quais medidores, uma vez que os agregadores são instalados nos postes. Para todos os agregadores é verificado quais medidores estão a uma distância menor ou igual ao alcance. Os que obedecem esta condição são representados por 1, caso contrário, por 0. A Figura 3 exemplifica este processo. Nela, os raios de cobertura são representados por circunferências tracejadas, os medidores por círculos e os postes por triângulos. A matriz de cobertura resultante também é exibida. Note que os medidores dentro da área de cobertura de um agregador são identificados pelo número 1, e os que estão fora por 0.

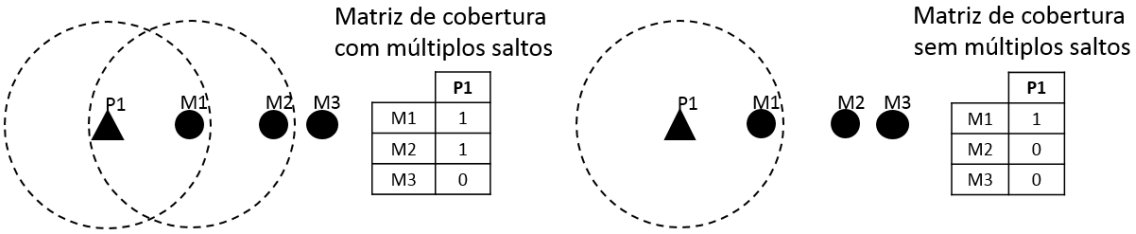


	P1	P2
M1	1	0
M2	0	0
M3	1	0
M4	1	1
M5	0	1

Figura 3. Criação da matriz de cobertura pela posição de medidores e agregadores.

Durante esta fase, o alcance dos agregadores é imutável e não considera obstáculos do meio. Por este motivo, é recomendado que este alcance seja obtido através de um modelo de propagação sem visada direta como, por exemplo, o Okumura-Hata [Hatay 1980].

De forma análoga, é possível construir a matriz de cobertura considerando múltiplos saltos. Medidores poderão atuar como nós intermediários, permitindo o alcance de dispositivos que antes seria impossível. Dado um poste e o conjunto de medidores alcançáveis a um salto, é considerado que este poste também alcança os vizinhos deste conjunto. O número máximo de saltos é variável. A Figura 4 exibe a matriz de cobertura da técnica com 2 saltos e com apenas 1 salto. Utilizando dois saltos, o medidor M1 é capaz de alcançar seu vizinho M2. Portanto, considera-se que o agregador P1 cobre M2 utilizando dois saltos.



	P1
M1	1
M2	1
M3	0

	P1
M1	1
M2	0
M3	0

Figura 4. Comparação da matriz de cobertura com e sem múltiplos saltos.

4.3. Construção da Matriz de Cobertura por Grade

O tempo de construção da matriz de cobertura está intimamente ligado à dimensão da região na qual postes e medidores estão instalados. Isto acontece pois cada poste deve ser analisado independentemente, verificando quais medidores um agregador que venha a ser instalado neste poste será capaz de cobrir. Contudo, verificar todos os medidores é desnecessário, visto que, dependendo da tecnologia de rede empregada é pouco provável que medidores muito distantes estarão dentro do alcance desse agregador. Esse problema se agrava à medida que a região analisada e a quantidade de medidores aumenta. Para evitar comparações desnecessárias, a adaptação proposta neste trabalho organiza medidores e postes em uma grade, que os divide em células de acordo com suas posições geográficas.

Uma grade é composta por várias células com tamanhos fixos. Durante a criação da matriz de cobertura, o poste analisado só verificará a conectividade de medidores que pertencem à sua célula e às células vizinhas. A Figura 5 exibe um exemplo de grade e a região que deve ser analisada mediante à escolha de uma célula. Os postes são representados por triângulos e os medidores por círculos preenchidos. A verificação das células vizinhas evita que um poste esteja no limite de uma célula e não analise um medidor alcançável que pertença a outra célula. Note que o poste da célula referência alcança um medidor da célula superior. É importante notar que caso o tamanho da célula seja muito maior que o alcance de um poste, maior será a quantidade de medidores analisados desnecessariamente. No pior caso, a grade contém uma única célula que engloba todos os postes e medidores. Consequentemente, para a grade com uma única célula, a quantidade de medidores analisados desnecessariamente será máxima, retornando ao problema original. Por exemplo, em uma região com 29002 medidores e 12140 postes e alcance de 50 m, a matriz de cobertura é criada em 2.6 s considerando uma grade de células de aproximadamente 100x100 m^2 e sem o uso da grade a criação demora 82,6 s. Neste cenário, o uso da grade acelerou o processo em aproximadamente 32 vezes.

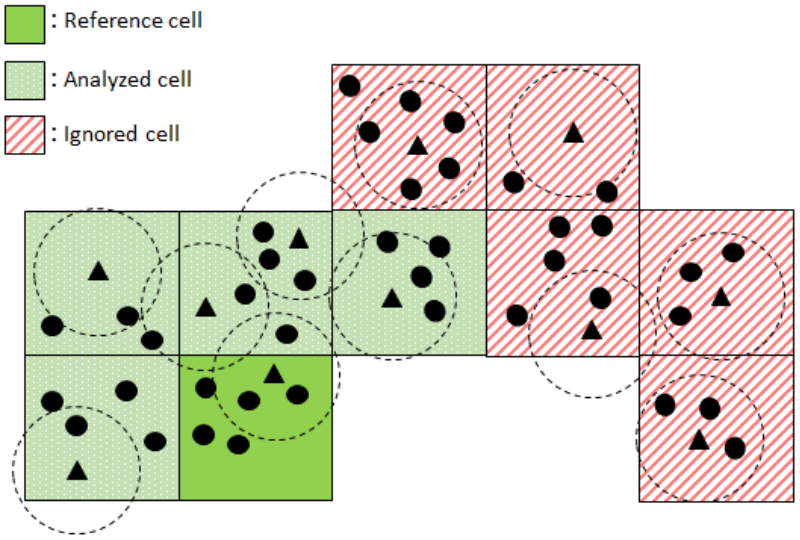


Figura 5. Uma grade e a região de alcance de uma célula.

Por outro lado, o tamanho da grade não pode ser menor que o alcance do agregador. Caso seja, há chance de um agregador em um dado poste alcançar um medidor de uma região não vizinha e consequentemente construir a matriz de cobertura incorretamente.

5. A Heurística Proposta

Duas abordagens podem ser utilizadas para resolver o SCP. A primeira envolve o uso de um *solver* de programação linear inteira, como por exemplo, o CPLEX [IBM 2009] e o GLPK [Makhorin 2008]. Esses métodos retornam as soluções exatas, mas ocupam grande quantidade de memória e exigem grande capacidade de processamento. Dependendo do tamanho e da complexidade do problema, a utilização de um *solver* é inviável. A segunda abordagem é utilizar uma heurística. As heurísticas não garantem que a solução

ótima será encontrada, mas são capazes de obter bons resultados utilizando menos recursos computacionais e em intervalos de tempo relativamente curtos.

O SCP modelado para o planejamento de agregadores em *smart grids* obedece um padrão geográfico, isto é, os medidores alcançáveis por um poste estão limitados a uma região ao redor deste. É improvável, por exemplo, que um poste alcance medidores muito distantes mas não alcance medidores que estejam próximos. Os *solvers* baseiam-se em características do problema para definir uma série de restrições, limitando o conjunto solução e, com isso, obtendo o resultado ótimo através de tentativas. Não cabe a este trabalho explicar o funcionamento de um *solver*. O problema formado a partir das características mencionadas são as mesmas de um SCP do tipo euclidiano. [Yelbay et al. 2012] detalham que SCPs do tipo euclidiano são de fácil resolução e que o avanço dos *solvers* ao longo dos anos permitiu que estas instâncias fossem resolvidas rapidamente. De fato, essa característica pode ser observada através do seguinte experimento computacional. A biblioteca online OR-Library [Beasley 1990] disponibiliza, para testes, inúmeras instâncias para diferentes problemas de otimização. A menor instância não euclidiana disponibilizada com 200 elementos (medidores) e 1000 subconjuntos (postes) foi submetida ao *solver* GLPK. Em 9 horas de execução a solução ainda não tinha sido descoberta. Uma instância de um cenário de *smart grid* (euclidiana), com posições de postes reais disponibilizadas pela Celesc (Centrais Elétricas de Santa Catarina), tendo 9997 medidores, 750 postes e agregadores com alcance máximo de 50 metros submetida ao mesmo *solver* foi resolvida em 1,9 s, desconsiderando o tempo de alocação de memória.

As características particulares do SCP formulado a partir de cenários de *smart grids*, portanto, podem ser processadas em intervalos curtos de tempo pelos *solvers* em virtude da disposição geográfica e do alcance limitado dos agregadores. No entanto, a quantidade de memória necessária e o tempo de alocação dessa quantidade de memória crescem conforme a dimensão do problema. Para instâncias grandes, é inútil se ter alta capacidade de processamento se a alta exigência de memória inviabiliza sua execução.

A heurística proposta aproveita-se das características apresentadas acima e do uso de grades para a construção da matriz. O problema é dividido em partes menores, que serão aplicadas ao método exato independentemente. Para cada célula forma-se um SCP a partir dos medidores desta célula e de postes desta e das células vizinhas. Os postes das células vizinhas são utilizados para se aumentar o grau de escolhas durante a execução do método exato. A união das soluções de todas as células compõe a solução para o problema inicial. Contudo, um mesmo poste pode fazer parte da solução de mais de uma célula. Por este motivo, caso a solução final contenha postes repetidos, considera-se que um único agregador deverá ser instalado nestes postes. A Figura 6 exibe o procedimento de divisão da heurística. O problema inicial indicado por P1 é aplicado a uma grade G, para cada célula desta grade forma-se um subproblema com os medidores e postes de uma célula acrescido dos postes de células vizinhas. A célula C1 gera o subproblema S1, a C2 gera o subproblema S2 e a C3 gera o subproblema S3. Cada subproblema é aplicado ao método exato. A união dos resultados de S1, S2 e S3 compõe a solução do problema P1.

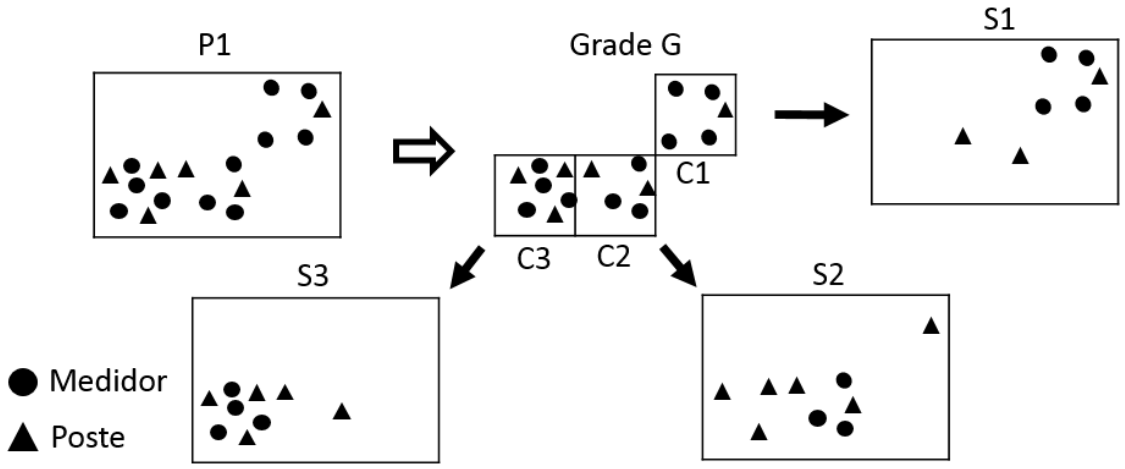


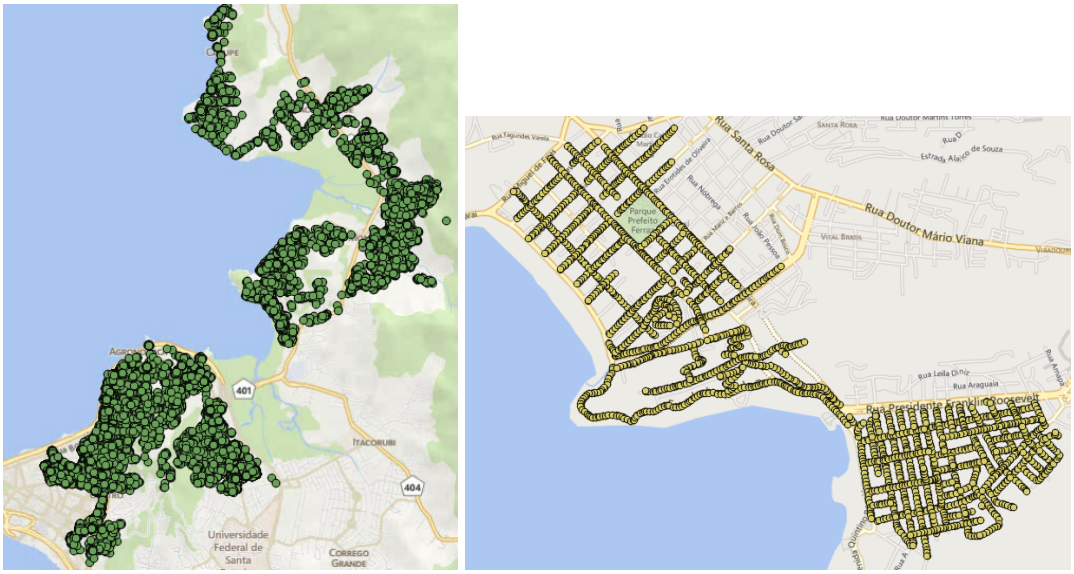
Figura 6. Processo de divisão da heurística proposta.

6. Resultados

6.1. Desempenho

O primeiro objetivo da avaliação é comparar o desempenho da heurística proposta com o do método exato baseado em programação linear inteira implementado pelo *solver* GLPK quanto ao tempo de execução, qualidade da solução (quantidade de agregadores utilizados) e exigência de memória. O objetivo do SCP é obter o menor número de agregadores necessário para se cobrir toda a vizinhança. Todos os métodos foram executados em uma única *thread* em um Intel Core i5 de 3 GHz com 8 GB de memória RAM no sistema operacional Windows 8.1. A primeira instância a ser analisada foi criada a partir de posições geográficas reais de postes de bairros da cidade de Florianópolis disponibilizados pela Celesc. Essa instância contém ao total 12140 postes e 29002 medidores distribuídos ao redor destes e é chamada de Floripa. A Figura 7(a) exibe esta região. A segunda instância foi gerada manualmente baseando-se nas posições de ruas e casas dos bairros de Icaraí e São Francisco na cidade de Niterói. Medidores foram distribuídos ao longo das ruas espaçados uns dos outros de 15 a 25 m, aproximadamente. A distância entre postes é de 30 a 50 metros aproximadamente. O intuito dessa forma de distribuição usada na segunda instância, chamada de Niterói, é simular uma possível distribuição de medidores e postes para analisar como o planejamento se comporta. As posições utilizadas não são reais. A Figura7(b) mostra esta região.

Para aumentar e diversificar a avaliação, as instâncias Floripa e Niterói são divididas em sub-instâncias. A instância Floripa foi dividida em 8 sub-instâncias de forma aditiva. Por exemplo, a segunda instância é composta pela primeira acrescida de novos medidores e postes da instância original, a terceira engloba a segunda e assim sucessivamente, até que a oitava sub-instância é, na verdade, a instância completa, contendo os 12140 postes e 29002 medidores. A instância Niterói foi dividida em duas sub-instâncias, a primeira contendo apenas um bairro e a segunda ambos. A Tabela 1 compara o tempo de execução total do método proposto, chamado de *Grid Planning* com diferentes dimensões de célula. A sigla GP significa *Grid Planning* e o número à direita a dimensão da célula. Por exemplo, GP 100 utiliza células de 100x100 m^2. O caso particular GP 10000 foi utilizado propositalmente para que contenha apenas uma única célula, representando a



(a) Floripa (b) Niterói

Figura 7. Disposição dos medidores para as instâncias Floripa e Niterói.

solução ótima. O GLPK foi escolhido como *solver* que será utilizado em cada célula. As sub-instâncias são identificadas pelo nome da instância a qual foi derivada seguido de sua dimensão (medidores x postes). O alcance de um agregador foi definido por 50 m e que medidores devem se conectar diretamente aos agregadores. T.E indica o tempo de execução em segundos. Q.S é a qualidade da solução dada em número de agregadores com o valor de *gap* (distância da melhor solução encontrada) entre parênteses. A sigla M.M.A indica a quantidade máxima de memória RAM alocada em megabytes.

Com dimensões maiores de célula, o método proposto possui maior informação da vizinhança como um todo e, consequentemente, é capaz de gerar melhores soluções. Por outro lado, o tamanho da célula está associado à quantidade de memória exigida e ao tempo de execução. Conforme mencionado, no caso do GP 10000, essa dimensão é tão extensa que a grade é composta por apenas uma célula, sendo equivalente ao método exato. Essa característica também pode ser observada em instâncias menores. Nas sub-instâncias Floripa com 1000, 2000 e 3000 medidores, o GP 1000 obteve resultados idênticos ao GP 10000, indicando que para estas sub-instâncias, uma única célula de 1000x1000 m^2 englobou todos os elementos.

Para a instância Floripa completa, o GP 10000 não foi capaz de obter o resultando, estourando a capacidade máxima de 8 GB de RAM. Essa mesma instância foi submetida a outro computador com 16 GB de RAM, com o intuito de se determinar o limite de memória exigido. Os 16 GB também foram esgotados.

Em contra partida, o GP 100 teve as piores qualidades de solução, apesar de exigir menos quantidade de memória e obter os menores tempos de execução (desconsiderando a primeira e segunda instâncias). O pior dos casos acontece na instância Floripa completa, onde o GP 100 selecionou 1380 postes contra os 955 (não exato) obtidos pelo GP 1000, caracterizando um *gap* de 45%.

Tabela 1. Comparação de desempenho entre os métodos.

Instância		GP 100	GP 550	GP 1000	GP 10000 (ótimo)
Floripa(1000x72)	T.E(s)	0,46	0,32	0,30	**0,29**
	Q.S(nº DAPs)	37 (42%)	27(4%)	**26**(0%)	**26**(0%)
	M.M.A(MB)	**1,00**	10,10	12,80	12,80
Floripa(2000x138)	T.E(s)	**0,92**	1,12	1,20	1,17
	Q.S(nº DAPs)	67(40%)	**48**(0%)	**48**(0%)	**48**(0%)
	M.M.A(MB)	**1,50**	18,50	45,90	45,90
Floripa(3000x209)	T.E(s)	**1,00**	2,20	2,52	2,57
	Q.S(nº DAPs)	97(31%)	76(3%)	**74**(0%)	**74**(0%)
	M.M.A(MB)	**1,60**	26,20	66,00	101,50
Floripa(4000x274)	T.E(s)	**1,76**	3,16	4,30	4,63
	Q.S(nº DAPs)	138(28%)	114(6%)	111(3%)	**107**(0%)
	M.M.A(MB)	**1,60**	44,50	93,00	174,00
Floripa(5000x354)	T.E(s)	**2,46**	4,32	6,80	7,04
	Q.S(nº DAPs)	181(32%)	147(7%)	144(5%)	**137**(0%)
	M.M.A(MB)	**1,20**	46,79	141,00	278,10
Floripa(10000x750)	T.E(s)	**5,5**	9,28	21,14	31,56
	Q.S(nº DAPs)	399(28%)	339(9%)	329(6%)	**311**(0%)
	M.M.A(MB)	**2,20**	48,50	220,20	1167,90
Floripa(15000x1872)	T.E(s)	**12,53**	20,50	45,91	158,32
	Q.S(nº DAPs)	846(26%)	714(7%)	691(3%)	**670**(0%)
	M.M.A(MB)	**3,30**	91,69	407,50	4340,00
Floripa(29002x12140)	T.E(s)	**43,00**	168,00	739,88	*
	Q.S(nº DAPs)	1380(45%)	1004(5%)	**955**(0%)	*
	M.M.A(MB)	**22,00**	1386,00	5322,10	*
Niterói(1576x453)	T.E(s)	**1,67**	2,54	2,8	2,94
	Q.S(nº DAPs)	155(44%)	120(11%)	109(1%)	108(0%)
	M.M.A(MB)	**0,70**	27,60	102,70	113,50
Niterói(3666x1030)	T.E(s)	**4,65**	5,95	9,89	15,43
	Q.S(nº DAPs)	409(46%)	304(9%)	280(0%)	280(0%)
	M.M.A(MB)	**0,70**	28,90	143,70	588,30

*O método não foi capaz de obter um resultado por falta de memória.

As mesmas instâncias foram submetidas a testes utilizando a técnica de múltiplos saltos, descrita na Seção 4. A Tabela 2 exibe os resultados assumindo o valor limite de 4 saltos. Os resultados mostram que o uso de múltiplos saltos reduz o número de agregadores exigidos para cobrir uma dada região. Para a instância completa Floripa, ocorre uma queda de 955 para 299 postes selecionados, o que significa uma redução de 656 agregadores nesta região. O uso de múltiplos saltos dificulta o problema, aumentando o tempo de execução dos métodos, contudo, pouco influencia na quantidade de memória necessária. Essa característica é evidente na instância Niterói(3666x1030), que sem múltiplos saltos demorou 15,43 s e utilizou 588,30 MB de memória, enquanto que com múltiplos saltos demorou aproximadamente 12 vezes mais tempo para ser resolvida, sendo que a quantidade máxima de memória subiu em apenas 16,6 MB. O limite de saltos deve ser ajustado conforme as características da região. Cenários urbanos, por exemplo, possuem inúmeras fontes de ruído e interferência, portanto diminuir a quantidade de saltos aparenta ser uma boa decisão. Opostamente, a livre propagação do sinal em campos rurais permite que o número de saltos seja maior.

6.2. Aumento de Cobertura pela Técnica de Múltiplos Saltos

Conforme exemplificado na Seção 4, durante a criação da matriz de cobertura de uma região, é possível que alguns medidores não sejam alcançáveis por estarem muito distantes de qualquer poste. Por isso, uma solução é empregar redes de múltiplos saltos.

Tabela 2. Comparação de desempenho entre os métodos com até 4 saltos.

Instância		GP 100	GP 550	GP 1000	GP 10000 (ótimo)
Floripa(1000x72)	T.E(s)	**0,58**	0,75	0,83	0,82
	Q.S(nº DAPs)	18 (200%)	7(16%)	**6**(0%)	**6**(0%)
	M.M.A(MB)	**1,10**	11,7	15,00	15,00
Floripa(2000x138)	T.E(s)	**1,27**	2,44	3,18	3,05
	Q.S(nº DAPs)	38(171%)	17(21%)	**14**(0%)	**14**(0%)
	M.M.A(MB)	**1,60**	19,80	51,00	51,00
Floripa(3000x209)	T.E(s)	**1,91**	3,99	4,82	6,12
	Q.S(nº DAPs)	60(200%)	23(15%)	23(15%)	**20**(0%)
	M.M.A(MB)	**1,80**	27,9	69,60	110,10
Floripa(4000x274)	T.E(s)	**2,70**	5,29	7,29	9,77
	Q.S(nº DAPs)	92(178%)	46(39%)	40(21%)	**33**(0%)
	M.M.A(MB)	**1,70**	48,70	100,30	185,10
Floripa(5000x354)	T.E(s)	**3,31**	6,92	10,81	14,65
	Q.S(nº DAPs)	117(179%)	55(31%)	48(14%)	**42**(0%)
	M.M.A(MB)	**1,60**	50,20	148,90	290,60
Floripa(10000x750)	T.E(s)	**7,34**	15,22	29,22	58,58
	Q.S(nº DAPs)	271(130%)	154(31%)	131(11%)	**118**(0%)
	M.M.A(MB)	**2,60**	51,90	228,50	1188,40
Floripa(15000x1872)	T.E(s)	**15,36**	31,19	60,51	272,77
	Q.S(nº DAPs)	602(92%)	383(22%)	347(10%)	**313**(0%)
	M.M.A(MB)	**4,50**	98,30	423,70	4384,50
Floripa(29002x12140)	T.E(s)	**104,73**	492,88	985,89	*
	Q.S(nº DAPs)	803(169%)	342(14%)	**299**(0%)	*
	M.M.A(MB)	**28,00**	1566,90	5726,60	*
Niterói(1576x453)	T.E(s)	**2,13**	3,25	4,85	4,55
	Q.S(nº DAPs)	82(412%)	22(38%)	18(13%)	**16**(0%)
	M.M.A(MB)	**0,80**	30,0	111,50	123,20
Niterói(3666x1030)	T.E(s)	**4,94**	6,86	11,85	184,07
	Q.S(nº DAPs)	234(350%)	70(35%)	59(13%)	**52**(0%)
	M.M.A(MB)	**0,80**	30,7	152,19	604,90

*O método não foi capaz de obter um resultado por falta de memória.

Para avaliar o emprego dessa solução, o segundo objetivo é comparar a quantidade de medidores alcançáveis ao se variar a quantidade máxima de saltos permitida quando se emprega a solução com múltiplos saltos. O alcance foi reduzido a 10 metros para que o número de medidores não alcançáveis seja mais evidente. O número de saltos variou de 1 (sem múltiplos saltos) a 6 saltos. A Tabela 3 exibe os resultados obtidos. Notavelmente o uso de múltiplos saltos aumenta a cobertura de uma rede, sobretudo nas sub-instâncias Floripa.

Tabela 3. Relação de medidores não alcançáveis e número de saltos máximo.

	Número máximo de saltos					
Instância	1	2	3	4	5	6
Floripa(1000x72)	164	8	7	7	7	7
Floripa(2000x138)	363	15	15	15	15	15
Floripa(3000x209)	526	28	28	28	28	28
Floripa(4000x274)	728	26	23	23	23	23
Floripa(5000x354)	882	42	40	40	40	40
Floripa(10000x750)	1809	93	82	81	81	81
Floripa(15000x1872)	2548	216	210	210	210	210
Floripa(29002x12140)	3201	502	486	486	486	486
Niterói(1576x453)	522	464	457	456	456	456
Niterói(3666x1030)	1299	1166	1147	1146	1146	1146

7. Conclusão e Trabalhos Futuros

Este trabalho adaptou o problema de planejamento em redes elétricas inteligentes para um problema conhecido na literatura e propôs um método para resolvê-lo. Embora o *Set Covering Problem* seja NP-Completo, as informações de cobertura sob a qual foi montado baseiam-se nas posições geográficas de medidores e postes e do alcance máximo de um agregador. Essa característica também permitiu a separação de elementos em células, utilizadas para reduzir o tempo necessário de criação da matriz de cobertura. Adicionalmente, as células fazem parte da heurística proposta, delimitando regiões que serão planejadas independentemente, permitindo a resolução de grandes instâncias. Os resultados mostram que solucionar cada célula independentemente reduz a quantidade de memória necessária e o tempo de execução. No entanto, o tamanho das células impacta diretamente no resultado, de forma que a qualidade da solução, tempo de processamento e quantidade de memória aumentam de acordo com a dimensão de uma célula.

As soluções apresentadas têm por objetivo determinar a menor quantidade possível de agregadores para cobrir uma região. Contudo, o valor de alcance de um agregador deve ser condizente com o cenário em que os agregadores serão instalados. Caso o alcance seja muito otimista, o número de agregadores obtido pela solução será menor que o de fato se precisa para cobrir a região. Caso o alcance seja muito pessimista, a solução conterá agregadores redundantes. Como trabalho futuro propõe-se que este alcance seja obtido por um modelo de propagação sem visada direta que considere aspectos da região a ser planejada como urbano, suburbano e rural [CEPT Administrations 2000]. Também propõe-se realizar aprimoramentos na heurística apresentada, aplicando métodos de busca local no resultado final. Futuramente, as soluções obtidas serão validadas sobre simulações no NS3.

Agradecimentos

Este trabalho é apoiado em parte por CNPq, CAPES, FAPERJ, TBE/ANEEL e Celesc/ANEEL.

Referências

Beasley, J. E. (1990). OR-Library: distributing test problems by electronic mail. *Journal of the operational research society*, pags. 1069–1072.

CEPT Administrations (2000). Monte-carlo simulation methodology for the use in sharing and compatibility studies between different radio services or systems. *ERC within the CEPT*.

EDX Wireless (2014). EDX SignalPro v8.2. `http://edx.com/products/edx-signalpro/`.

Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., e Goh, M. (2012). Covering problems in facility location: A review. *Computers & Industrial Engineering*, 62(1):368–407.

Guimarães, P. H. V., Murillo, A., Andreoni, M., Mattos, D. M. F., Ferraz, L. H. G., Pinto, F. A. V., Costa, L. H. M. K., e Duarte, O. C. M. B. (2013). Comunicação em redes elétricas inteligentes: Eficiência, confiabilidade, segurança e escalabilidade. *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC 2013*, pags. 101–165.

Hatay, M. (1980). Empirical formula for propagation loss in land mobile radio services. *IEEE Transactions on Vehicular Technology*, 29(3):317–325.

IBM (2009). CPLEX optimizer: High-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming.

Karp, R. M. (1972). *Reducibility among combinatorial problems*. Springer.

Leon, G. (2011). Smart planning for smart grid AMI mesh networks. Relatório técnico, EDX Wireless.

Makhorin, A. (2008). GLPK (GNU linear programming kit).

Okabayashi, V. H. (2014). Seleção dinâmica de gateway para redes elétrica inteligentes. Dissertação de Mestrado, Universidade Federal Fluminense.

Rolim, G. (2014). Vídeo *A Network Planning System for Smart Grid* vencedor da competição de estudantes do SmartGridComm 2014. `http://sgc2014.ieee-smartgridcomm.org/student-competition`.

Rolim, G., Albuquerque, C., Moraes, I., Bettiol, A., Carniato, A., Passos, L. F., Homma, R. Z., e Molina, F. H. (2014). Sistema de Posicionamento de Agregadores de Dados em Redes Elétricas Inteligentes. Aceito para publicação em XIII ERIAC - Décimo Terceiro Encontro Regional Iberoamericano do Cigré.

Souza, G., Vieira, F., Lima, C., Junior, G., Castro, M., e Araujo, S. (2013). Optimal positioning of GPRS concentrators for minimizing node hops in smart grids considering routing in mesh networks. Em *IEEE PES Conference On Innovative Smart Grid Technologies Latin America (ISGT LA)*, pags. 1–7.

Souza, Gustavo B. de C., Viera, F. H. T., Cardoso, A. A., Lima, C. R., de Deus Júnior, G. A., de Castro, M. S., e de Araújo, S. G. (2013). Utilização de programação linear inteira para posicionamento de concentradores GPRS em redes smart grid. *Simpósio Brasileiro de Automação Inteligente*, pags. 1–6.

Toregas, C., Swain, R., ReVelle, C., e Bergman, L. (1971). The location of emergency service facilities. *Operations Research*, 19(6):1363–1373.

Wenpeng, L. (2009). Advanced metering infrastructure. *Southern Power System Technology*, 3(2):6–10.

Yelbay, B., Birbil, Ş. İ., e Bülbül, K. (2012). The set covering problem revisited: an empirical study of the value of dual information. *Optimization Online*.

Um Modelo Autônomo de Processamento de Consultas Espaciais para Redes de Sensores Urbanas

Marcos A. Carrero[1], Rone I. da Silva[2], Carmem S. Hara[1], Aldri L. dos Santos[1]

[1]DINF – Universidade Federal do Paraná – UFPR – Paraná, Brasil

[2]DTECH – Universidade Federal de São João del-Rei – UFSJ – Minas Gerais, Brasil

{macarrero,carmem,aldri}@inf.ufpr.br, rone@ufsj.edu.br

Abstract. *Wireless Sensor Networks (WSN) in urban environments manage a large amount of sensoring data. The deployment of spatial query processing in a decentralized and autonomous large-scale WSN is a major challenge due to the network resources constraints. This paper proposes ASQPM, a scalable and autonomous model for data storage and spatial query processing. Scalability is provided by grouping sensors into clusters based on the spatial similarity of their readings. The query processing efficiency relies on the concept of repositories, which are regions in the monitored area that concentrate information, storing the readings of a* set *of clusters. The experimental results show that it is more effective for query processing than classical approaches.*

Resumo. *As Redes de Sensores sem Fio (RSSFs) urbanas lidam com grande quantidade de informações coletadas. O processamento de consultas espaciais em RSSFs de larga escala é um grande desafio devido às limitações de recursos da rede. Este trabalho propõe o ASQPM, um modelo escalável e autônomo para o armazenamento de dados e processamento de consultas espaciais. A escalabilidade resulta da estratégia de formação de agrupamentos de sensores que possuem similaridade espacial de leituras. A eficiência da consulta é determinada pelo conceito de repositório, que são regiões de concentração de dados de um conjunto de agrupamentos. Os resultados experimentais mostram que o seu processamento de consulta é mais eficaz do que as abordagens clássicas.*

1. Introdução

As Redes de Sensores Sem Fio (RSSFs) oferecem uma infra-estrutura para o desenvolvimento de aplicações em várias áreas de interesse. São exemplos de aplicações o monitoramento do ambiente, sistemas militares, de vigilância e de saúde [Yick et al. 2008]. Em geral, os sensores possuem recursos limitados como baixa largura de banda, comunicação de curto alcance, armazenamento limitado e processamento. Em ambientes urbanos, os sensores podem ser densamente distribuídos a fim de coletar diferentes atributos ambientais tais como temperatura, pressão, umidade, luminosidade e poluição. Além disso, os dados coletados podem ser armazenados nos próprios sensores, proporcionando o desenvolvimento de aplicações que executam consultas na rede, sem depender de um servidor central. Em particular, em RSSFs de grande escala, manter todo o processamento centralizado aumenta o custo de comunicação [Can and Demirbas 2013], sendo menos escalável que abordagens descentralizadas [Coman et al. 2007].

As consultas processadas por uma RSSF podem ser classificadas em consultas espaciais e por valor. As consultas espaciais têm como objetivo obter os valores coletados por sensores dentro de uma área geográfica de interesse. As consultas por valor, por outro lado, determinam quais sensores coletaram dados dentro de um intervalo de valores. Neste artigo o foco é sobre consultas espaciais. A disseminação de consultas é uma tarefa difícil devido ao alto custo de comunicação e de restrições dos recursos da rede. No entanto, explorando algumas características encontradas nas RSSFs é possível reduzir a sobrecarga na comunicação [Le et al. 2008, Cheng et al. 2011]. Em algumas aplicações, a correlação espaço-temporal presente nas leituras de sensores próximos possibilita reduzir o número de transmissões na rede. No caso da correlação espacial, leituras de sensores próximos tendem a ser muito similares nos seus valores. Em relação à segunda característica, leituras consecutivas tendem a ser muito próximas no tempo [Vuran et al. 2004]. Explorando-se a correlação espacial, que é o foco deste trabalho, é possível alcançar um requisito muito desejável que é a escalabilidade, organizando em grupos (*clusters*) sensores que apresentarem leituras similares em seus valores [Hung et al. 2012].

Além da característica espacial, a gestão do armazenamento de dados em RSSFs possui impacto significativo no uso de recursos e no desempenho do processamento de consultas [Yu et al. 2010]. Em geral, os dados coletados podem ser armazenados localmente no próprio sensor, em uma base de dados externa, ou em *repositórios* de dados distribuídos na própria rede. Embora a melhor escolha dependa do contexto da aplicação, nota-se que o modelo de armazenamento de dados em *repositórios* oferece uma abordagem interessante entre o custo da consulta e o custo do armazenamento dos dados [Xie et al. 2014]. Logo, a eficiência no processamento de consultas baseia-se no estabelecimento de *repositórios* na rede, reduzindo-se o número de encaminhamento de consultas, isto é, de saltos na rede, para se obter o resultado desejado.

Inspirados por estes desafios, trabalhos recentes que lidam com o agrupamento de sensores com base na similaridade de dados têm sido propostos, como o DCSSC [Le et al. 2008], o SIDS [Furlaneto et al. 2012] e o DCASC [Ma et al. 2011]. Nos trabalhos DCSSC e DCASC, sensores com alta correlação em suas leituras são agrupados e um sensor líder é eleito como representante do grupo. No entanto, nenhum deles especifica um modelo para disseminação de consultas espaciais e nem a criação de *repositórios* na rede para reduzir o número de acessos aos agrupamentos, de modo que o processamento da consulta em uma rede de grande escala pode ser cara. Além disso, outra desvantagem do DCSSC é que as consultas são iniciadas a partir de um único ponto de entrada externo. Por outro lado, o SIDS combina a formação de agrupamentos com base na similaridade espacial dos dados com a criação de *repositórios*, porém não é um modelo autônomo pois depende de uma entidade externa para coordenar o processo de agrupamento.

Este trabalho propõe um modelo autônomo de processamento de consultas espaciais para RSSFs urbanas, chamado ASQPM (*Autonomous Spatial Query Processing Model*). O ASQPM é um modelo hierárquico e distribuído onde cada sensor armazena localmente os dados de monitoramento, e sensores com alta correlação espacial são organizados em grupos. No nível de agrupamento, um sensor é eleito para ser o representante do grupo e sua leitura sensoriada representa a leitura do seu agrupamento. No nível de *repositório*, sensores específicos que servem como centro de dados armazenam informações de um conjunto de agrupamentos distintos, reduzindo o custo total de comunicação no pro-

cessamento de consultas. O modelo é autônomo porque a própria rede atua na formação de agrupamentos, na escolha do representante do grupo e na definição de *repositórios*, sem depender de uma entidade externa central. Em nosso entendimento, este é o primeiro modelo autônomo que combina similaridade espacial de leituras com armazenamento distribuído em *repositórios* para reduzir a comunicação na rede no processamento de consultas espaciais no âmbito de RSSFs urbanas. Simulações mostram que o modelo reduz consideravelmente o tempo de processamento das consultas espaciais.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve o funcionamento do modelo ASQPM. A Seção 4 mostra os resultados obtidos na avaliação de desempenho e a conclusão e trabalhos futuros são discutidos na Seção 5.

2. Trabalhos Relacionados

Diferentes abordagens têm sido propostas para tratar o armazenamento de dados, a construção de agrupamentos e o processamento de requisições espaciais para RSSFs. O gerenciamento dos dados na rede possibilita o desenvolvimento de aplicativos eficientes de processamento de consultas espaciais. Além disso, abordagens clássicas frequentemente usadas para tratar a escalabilidade das aplicações são as técnicas de agrupamentos de sensores. Nas redes que apresentam correlação no dado sensoriado, os sensores podem ser agrupados com base na similaridade de suas leituras. Contudo, investigando a revisão literária sobre requisição de dados espaciais apresentada em [Da Silva et al. 2014], verificou-se que nenhum trabalho utiliza agrupamentos por similaridade de dados na rede a fim de melhorar o desempenho do processamento de consultas.

Estratégias usadas por CAG [Yoon and Shahabi 2007], DCSSC [Le et al. 2008] e DCASC [Ma et al. 2011] descrevem mecanismos que lidam com a característica da similaridade de dados sensoriados. Em geral, nestes trabalhos os sensores são agrupados e há seleção de um líder como representante do grupo, de modo que os sistemas sejam escaláveis. Contudo, nenhumas destas abordagens são totalmente autônomas e eficientes no processamento de consultas espaciais. O CAG precisa constantemente reconstruir os agrupamentos da rede para estabelecer novas consultas. O DCSSC, por outro lado, depende da estação base para gerenciar a criação dos agrupamentos. O DCASC apresenta um algoritmo para construção de agrupamentos de dados sensoriados, no qual este trabalho se inspirou, porém o DCASC não oferece suporte para consultas espaciais.

Um modelo de armazenamento de dados que leva em conta a similaridade de leituras é descrita por SIDS [Furlaneto et al. 2012]. Ele estabelece um mecanismo de agrupamentos baseados na similaridade espacial dos dados, um esquema para eleição de líderes e uma estrutura de indexação para evitar inundações. Além disso, locais de concentração de informação, denominados de *repositórios* de dados são formados na rede, consistindo de pelo menos dois sensores líderes em região de borda. No entanto, o SIDS é um modelo centralizado que depende da estação base para criar os agrupamentos.

Entre os algoritmos de consultas espaciais, o IBIS [da Silva et al. 2011] descreve uma proposta para RSSFs eficiente na redução do consumo de energia da rede. O IBIS é um mecanismo para processamento de requisições espaciais irregulares, baseado na criação de itinerários. Durante a requisição de uma consulta espacial, ele cria um itinerário para encaminhar e agregar os dados sensoriados da região de interesse. No en-

tanto, o IBIS não leva em conta a similaridade de dados e a formação de agrupamentos de sensores. Assim, um modelo de processamento de consultas espaciais que atenda os requisitos encontrados em RSSFs urbanas se faz necessário. Tal modelo deve ser capaz de combinar estratégias de agrupamentos por similaridade de dados e de formação de *repositórios*, adaptando-as para dar suporte a uma rede com alta densidade dos sensores, provendo escalabilidade e mantendo completamente a autonomia da rede.

3. O Modelo ASQPM

Esta seção descreve um modelo hierárquico e distribuído para realizar o processamento de consultas espaciais eficientes em redes que apresentam similaridade espacial nas leituras dos sensores. Este modelo, chamado ASQPM (*Autonomous Spatial Query Processing Model*), organiza em grupos sensores próximos que possuam similaridade em suas leituras. Uma vez definidos os agrupamentos, inicia-se o processo de seleção de sensores denominados de *repositórios*, responsáveis por armazenar informações de um conjunto de agrupamentos distintos. A seguir são detalhados o modelo de rede, os algoritmos de correlação espacial, de formação agrupamentos e de escolha de *repositórios*, bem como o modo de processamento das consultas espaciais.

3.1. Modelo de Rede

Uma RSSF é representada como um grafo $G = (V, L)$, onde $V = \{s_1, \ldots, s_n\}$ é um conjunto de sensores dispersos sobre a área monitorada M e L é o conjunto de ligações entre pares de sensores tal que $(s_i, s_j) \in L$ se s_i e s_j estão dentro do raio de comunicação um do outro. Diz-se que a distância entre s_i e s_j é de um *salto* e que s_i e s_j são *vizinhos*. A comunicação entre dois sensores quaisquer requer a existência de um caminho de roteamento $R = \{(s_1, s_2), (s_2, s_3), \ldots (s_{n-1}, s_n)\}$ tal que s_1 é o sensor que origina a mensagem e s_n é o seu destino final. Portanto, a comunicação em RSSFs depende da cooperação do repasse de mensagens na rede, e a escolha da rota R é tarefa do protocolo de roteamento. Neste trabalho, é assumido que os sensores são estáticos e, portanto, possuem um posicionamento geográfico fixo. Assume-se também que cada sensor realiza o sensoriamento de mais de um atributo do ambiente. As leituras de um sensor s são representadas por uma tupla $X = (x_1, x_2, \ldots, x_n)$, na qual cada x_i corresponde a um tipo de atributo de dado sensoriado como temperatura, umidade, luminosidade e poluição atmosférica.

Como exemplo, considere uma RSSF urbana monitorando diferentes valores de atributos ambientais, tais como temperatura, umidade, luminosidade e poluição do ar (CO, CO_2). A Figura 1(a) mostra os sensores densamente espalhados sobre algumas regiões da cidade, como parques, jardins, ruas e avenidas. Em cenários complexos, como o cenário urbano, se faz necessário trabalhar com uma granulação fina no monitoramento do ambiente [Thepvilojanapong et al. 2010]. O estudo mostrou que vários fatores ambientais influenciam na variação da temperatura, tais como a presença de árvores, a largura de estradas e diferenças entre as regiões geográficas. Além disso, devido à topografia irregular da cidade, verificam-se diferentes índices de concentração de poluentes mesmo em lados opostos da rua [Resch et al. 2009]. Assim, justifica-se o uso de uma RSSF densa como forma adequada de monitorar tal ambiente complexo [Muller et al. 2013] .

Neste trabalho, sensores próximos são agrupados com base na similaridade espacial dos dados, como ilustra a Figura 1(b). Cada agrupamento possui um sensor líder,

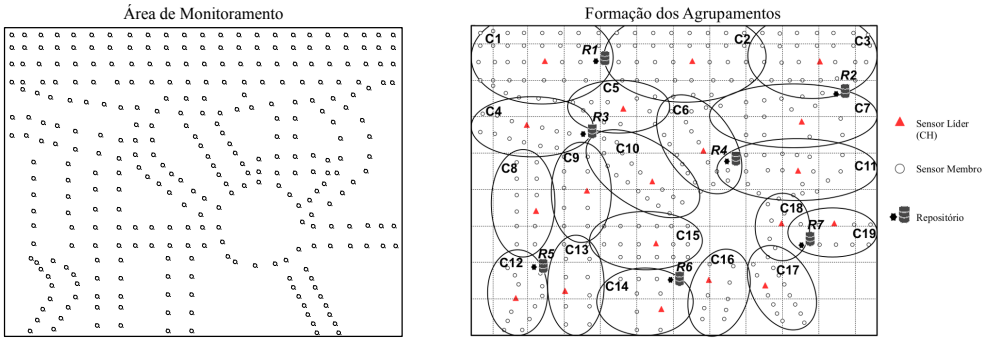


Figura 1. Visão geral do cenário e do modelo ASQPM

que é responsável pelo armazenamento de informações relevantes sobre os demais sensores do grupo. Os agrupamentos são definidos em áreas geográficas contíguas. Os dados amostrados dentro de cada agrupamento apresentam alta correlação entre os seus membros. Portanto, com base no entendimento acima, apenas os dados coletados pelo líder são relevantes durante o processamento de consultas, não necessitando das informações coletadas pelos demais membros. Assim, a sobrecarga no processamento e a comunicação dentro do agrupamento diminuem, resultando em menos uso de recursos.

A seleção do líder foi inspirada pelo método de correlação espacial apresentado em [Ma et al. 2011]. Lembre-se que nós sensores monitoram uma série de informações sobre o ambiente, onde $X = (x_1, x_2, \ldots, x_n)$ denota as leituras do sensor s_i, $Y = (y_1, y_2, \ldots, y_n)$ denota as leituras do sensor s_j e $N(i)$ denota o conjunto de vizinhos que estão a um *salto* de distância do sensor s_i. Portanto, o líder selecionado será aquele que possuir alta correlação entre suas leituras e as leituras de seus sensores vizinhos. O processo da seleção do líder se faz em várias etapas. Na primeira equação, a distância euclidiana entre as leituras do sensor s_i em relação ao sensor s_j é calculada como

$$d_{ij} = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \ldots + |x_n - y_n|^2} \tag{1}$$

Então, a média das distâncias $\overline{d_i}$, entre s_i e seus vizinhos, é dada por

$$\overline{d_i} = \frac{1}{|N(i)|} \sum_{j \in N(i)} d_{ij} \tag{2}$$

Além disso, o desvio médio absoluto de d_i é dado por

$$D(d_i) = \frac{1}{|N(i)|} \sum_{j \in N(i)} (d_{ij} - \overline{d_i})^2 \tag{3}$$

Portanto, o peso da correlação espacial $w(s_i)$ $(0 \leq w(s_i) \leq 1)$ de s_i é dado por

$$w(s_i) = \frac{\left[\sum_{j \in N(i)} |d_{ij} - \overline{d_i}|\right]^2}{|N(i)|^2 D(d_{ij})} = \frac{\left[\sum_{j \in N(i)} |d_{ij} - \overline{d_i}|\right]^2}{|N(i)| \sum_{j \in N(i)} (d_{ij} - \overline{d_i})^2} \tag{4}$$

De acordo com a equação acima, cada sensor s_i calcula um peso $w(s_i)$ que indica o quão correlacionados estão suas leituras com relação às leituras de seus vizinhos. Grandes valores de $w(s_i)$ indicam que as leituras de s_i e as leituras de seus vizinhos tendem a ser

altamente correlacionadas. De fato, um sensor s_i será selecionado como líder com base em um limite τ definido pelo usuário. Assim, a seleção do líder ocorre se $w(s_i) \geq \tau$. Intuitivamente, em uma vizinhança, o nó com maior peso será eleito como líder de um grupo de sensores que possuem similaridade numérica de leituras.

No modelo ASQPM existem três níveis de componentes: o sensor coletor s, o agrupamento h ao qual s pertence e o *repositório* r no qual os dados do agrupamento h são armazenados. No nível inferior, cada sensor s_i deve armazenar sua leitura atual, o peso da correlação $w(s_i)$, as leituras de seus vizinhos $readingsOfNeighbors(s_i)$, uma lista de seus vizinhos $N(i)$ e o líder do agrupamento ao qual ele pertence $CH(s_i)$. Assume-se que o mecanismo de disseminação de consulta depende de um protocolo de roteamento geográfico, de modo que cada sensor tem que armazenar sua própria informação geográfica $position(s_i)$. No nível superior, formado pelos agrupamentos e seus líderes, um sensor líder h armazena sua leitura atual, a posição geográfica dos membros do seu agrupamento $GC(h)$ (Geografia do Cluster) e o *repositório* no qual seus dados são armazenados $R(h)$. Observe que a partir do conjunto *GC* é possível determinar o menor retângulo delimitador que contém todos os sensores de um agrupamento $MBR(h)$. No nível mais alto, definido pelos *repositórios*, um sensor r estabelecido como *repositório* armazena as leituras de um conjunto de líderes, informações sobre as posições geográficas dos seus sensores GR (Geografia do Repositório) e uma lista dos *repositórios* mais próximos $knownRepo(r)$. É importante observar que um mesmo sensor pode assumir diversos papéis (*role*) simultaneamente, ou seja, além de um membro de um agrupamento, ele pode ser um líder (CH), um líder isolado (ICH) e/ou um *repositório* (R). Um líder é aquele que possui membros associados a ele. Por outro lado, um líder isolado não possui membros associados. O Algoritmo 1 mostra de forma detalhada o cálculo do peso da correlação espacial e da formação de agrupamentos no ASQPM.

Inicialmente, cada sensor envia a mensagem "SEND_READINGS" por inundação na rede, informando sua leitura atual *(l.6)*. Cada sensor s_i que recebe a mensagem armazena as leituras de seus vizinhos *(l.15-17)*. Após certo período de tempo Γ, cada sensor s_i executa a função "CALCULATE_WEIGHT" *(l.7-8)* para calcular o peso de sua correlação espacial $w(s_i)$, informando qual o limite (*threshold*) definido pelo usuário. A seleção do líder (CH) ocorre quando o peso $w(s_i)$, calculado pela função $getWeight(s_i)$, for maior do que o *threshold* informado. Em seguida, o CH eleito anuncia esta decisão para os demais sensores *(l.20-23)*. Ao receberem mensagens de anúncio, os sensores armazenam os possíveis candidatos a CH ($candidateCHs$) *(l.38)*. Se $w(s_i)$ for menor que o *threshold* informado, o sensor vira membro e espera por anúncios dos líderes *(l.24-25)*. Em seguida, o sensor membro escolhe como CH, dentre os sensores armazenados como possíveis candidatos ($candidateCHs$), aquele que tiver a leitura mais parecida com a sua, e então envia uma mensagem de anúncio para o CH eleito como seu líder *(l.26-27)*. O CH ao receber o ACK de associação, armazena a posição dos sensores membros $GC(s_i)$ *(l.32-34)*, possibilitando ao líder calcular qual a MBR do agrupamento. Por fim, o CH que não recebeu nenhuma mensagem de associação de um membro, tem seu papel alterado para líder isolado (ICH) *(l.10-12)*.

Dado que os sensores formam agrupamentos com base em sua localização espacial e similaridade em suas leituras, resta definir a estrutura do nível de *repositórios*. O *repositório* tem por finalidade minimizar o número de *saltos* durante o processamento de

Algoritmo 1 Peso da Correlação Espacial e Agrupamento

```
1: procedure GROUPING
2:     CH(s_i) ← s_i
3:     role[CH](s_i) ← FALSE
4:     role[R](s_i) ← FALSE
5:     readingsOfNeighbors(s_i) ← {}
6:     candidateCHs(s_i) ← {}
7:     GC(s_i) ← {}
8:
9:     broadcast('SEND_READINGS', currentReadings())
10:    WAIT(Γ time units)                                          ▷ Γ < Δ
11:    CALCULATE_WEIGHT(Threshold)
12:    WAIT(Δ time units)
13:    if (GC(s_i) = {}) then
14:        role[ICH](s_i) ← TRUE                                   ▷ Cluster-head alone
15:    end if
16: end procedure
17:
18: procedure RECEIVING('SEND_READINGS', NeighborReadings) by s_i
19:    readingsOfNeighbors(s_i) ← readingsOfNeighbors(s_i) ∪ {neighborReadings}
20: end procedure
21:
22: procedure CALCULATE_WEIGHT(Thresold) by s_i
23:    w(s_i) ← getWeight(readingsOfNeighbors(s_i))
24:    if (w(s_i) ≥ threshold) then
25:        role[CH](s_i) ← TRUE
26:        broadcast('CH_ANNOUNCEMENT', s_i)
27:    else
28:        WAIT(Γ time units) for CH_ANNOUNCEMENT
29:        CH(s_i) ← getCHWithSimilarReadings(candidateCHs(s_i))
30:        send ACK('CH_ANNOUNCEMENT', CH(s_i), position(s_i))
31:    end if
32: end procedure
33:
34: procedure RECEIVING('ACK_CH_ANNOUNCEMENT', h, pos) by s_i
35:    if role[CH](s_i) = TRUE then
36:        GC(s_i) ← GC(s_i) ∪ {pos}
37:    end if
38: end procedure
39:
40: procedure RECEIVING('CH_ANNOUNCEMENT', h) by s_i
41:    candidateCHs(s_i) ← candidateCHs(s_i) ∪ {h}
42: end procedure
```

consultas através da agregação de informações de líderes que estão próximos. Portanto, eles são os responsáveis por armazenar as leituras de um grupo de líderes tal que somente os dados dos *repositórios* serão utilizados durante o processamento de consultas espaciais. Intuitivamente, visto que os líderes representam dados de sensoriamento de cada membro do agrupamento, um *repositório* pode atuar como um centro de dados para responder a consultas referentes a quaisquer um dos agrupamentos que o compõem. O Algoritmo 2 apresenta a estratégia de escolha do *repositório*.

Inicialmente, a função *NeighboursInDistinctClusters* (s_i) *(l.3)* calcula o número de vizinhos de s_i que pertencem a agrupamentos distintos. O objetivo é procurar sensores que se encontram em regiões de borda para escolher como *repositório* aqueles que podem concentrar o maior número de agrupamentos. Assim, a seleção do sensor *repositório* ocorre se o número de agrupamentos em sua vizinhança for maior que um determinado patamar (*threshold*) definido pelo usuário *(l.4)*. Em seguida, o sensor escolhido como *repositório* anuncia esta decisão para os demais sensores *(l.5)*. Os sensores que recebem os anúncios dos *repositórios* armazenam cada *repositório* anunciado em um conjunto *knownRepo (l.20)*. Além disso, as mensagens de anúncio somente serão retransmitidas se a distância de s_i para o sensor r que enviou o anúncio for menor que a menor distância de algum *repositório* já conhecido *(l.21-23)*. Após aguardar certo período Δ de tempo, o

Algoritmo 2 Escolha do Repositório

```
1: procedure REPOSITORY_ELECTION(Threshold)
2:     knownRepo(s_i) ← {}
3:     if (NeighboursInDistinctClusters(s_i) > threshold) then
4:         role[R](s_i) ← TRUE
5:         broadcast('REPOSITORY_ANNOUCEMENT', s_i)
6:         WAIT(Δ time units)
7:         for all r in knownRepo(s_i) do
8:             send('MBR_ANNOUNCEMENT' ,s_i, r)
9:         end for
10:    else
11:        WAIT(Γ time units)
12:        if role[CH](s_i) = TRUE then
13:            R(s_i) ← getMinDistance(s_i, knownRepo(s_i))
14:            send('CH_JOIN_REPOSITORY', R(s_i), GC(s_i))
15:        end if
16:    end if
17: end procedure
18:
19: procedure RECEIVING('REPOSITORY_ANNOUCEMENT', r) by s_i
20:    knownRepo(s_i) ← knownRepo(s_i) ∪ {r}
21:    if knownRepo(s_i) = {} OR distance(s_i, r) < distance(s_i, getMinDistance(s_i, knownRepo(s_i))) then
22:        broadcast('REPOSITORY_ANNOUCEMENT', r)
23:    end if
24: end procedure
25:
26: procedure RECEIVING('CH_JOIN_REPOSITORY', r, gc) by s_i
27:    if R(s_i) = r then
28:        GR(s_i) ← GR(s_i) ∪ gc
29:    else
30:        forward('CH_JOIN_REPOSITORY', r, gc)
31:    end if
32: end procedure
33:
34: procedure RECEIVING('MBR_ANNOUNCEMENT', s_j, r) by s_i
35:    if R(s_i) = r then
36:        knownRepo(s_i) ← knownRepo(s_i) ∪ {s_j}
37:    else
38:        forward('MBR_ANNOUCEMENT', s_j, r)
39:    end if
40: end procedure
```

repositório se anuncia para os seus *repositórios* membros (*knownRepo*) *(l.6-9)*. O *repositório* membro que recebe a mensagem atualiza seu conjunto de *repositórios* conhecidos *(l.35-36)*. Mantendo um conjunto de *repositórios* conhecidos, um *repositório* consegue calcular qual a área de cobertura (MBR) de seus vizinhos. Por outro lado, um *repositório* simplesmente repassa uma mensagem que não foi destinada a ele *(l.38)*. Após certo período Γ de tempo, um CH associa como *repositório* aquele que tiver a menor distância dentre os *repositórios* conhecidos *(l.11-13)*. Em seguida, o CH envia uma mensagem de associação para o *repositório* informando sua MBR ($GC(s_i)$) *(l.14)*. O *repositório* de destino da mensagem atualiza sua MBR ($GR(s_i)$) com a informação da MBR enviada pelo CH *(l.27-28)*. Caso contrário, o sensor retransmite a mensagem para o *repositório* de destino *(l.30)*.

Como exemplo, considere o cenário ilustrado pela Figura 2, no qual o *repositório* $R3$ está em uma região de borda compreendida pelos agrupamentos $C4$, $C5$, $C9$ e $C10$, enquanto o *repositório* $R4$ está em uma vizinhança composta pelos agrupamentos $C6$, $C7$ e $C11$. Intuitivamente, uma vez que os *repositórios* armazenam informações sobre mais de um agrupamento, espera-se que o número de *saltos* realizados durante o processamento da consulta será menor.

As técnicas utilizadas pelo modelo de armazenamento distribuído do ASQPM apresentam um bom trade-off entre os custos de consulta e de armazenamento. Sem a

definição de agrupamentos ou *repositórios*, seria preciso inundar a rede com consultas para se obter o resultado desejado, o que é impraticável em redes de grande escala. Embora não detalhado neste artigo, os agrupamentos, CHs e *repositórios* são modificados de forma autônoma e similar aos algoritmos apresentados, sempre que a correlação espacial de um CH deixar de satisfazer o *threshold* pré-estabelecido.

3.2. Processamento de Consultas Espaciais

O processamento da consulta do ASQPM funciona da seguinte forma: qualquer sensor da rede pode iniciar uma consulta espacial, desta forma o modelo não se limita a um único ponto de entrada de consulta. O destino da consulta refere-se à região geográfica de interesse e o resultado refere-se às leituras coletadas por sensores nesta região. Observe que no modelo ASQPM as leituras dos líderes de agrupamentos representam as leituras de sua região geográfica e que *repositórios* armazenam dados sobre um conjunto de líderes. O protocolo de roteamento geográfico adotado é o GPSR [Karp and Kung 2000] para a fase de encaminhamento e resposta da consulta. Na fase de encaminhamento, o sensor que recebe a requisição de consulta encaminha o pedido para um sensor membro cuja região de cobertura se sobrepõe a região delimitada pela consulta. Em seguida, o sensor membro encaminha a mensagem para seu líder, que por sua vez envia a requisição de consulta para o *repositório*.

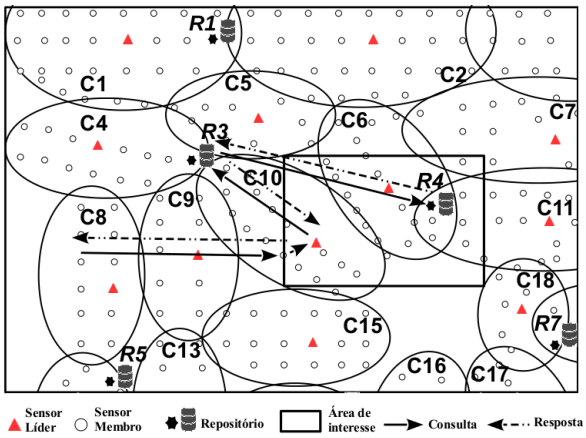


Figura 2. Posicionamento do repositório e processamento de consultas

A Figura 2 ilustra uma consulta espacial, onde neste exemplo, a requisição de consulta quer obter dados de uma região compreendida pelos agrupamentos $C6$, $C7$, $C10$ e $C11$. Suponha como ponto de entrada da consulta um sensor do agrupamento $C8$. O encaminhamento da consulta se faz da seguinte forma: inicialmente encaminha-se a requisição da consulta em direção à região de interesse até encontrar um sensor membro em $C10$ que está dentro desta região. Ao receber a requisição de consulta, o sensor membro a encaminha para o líder de $C10$, que por sua vez encaminha a requisição para o *repositório* $R3$. Assim, se a área de $R3$ (dada por GR) cobrir inteiramente a área de interesse da consulta, o processamento da consulta termina com o encaminhamento do resultado para o seu ponto de entrada.

Contudo, se a área de $R3$ não cobrir inteiramente a área de interesse da consulta, o encaminhamento da consulta direciona a requisição para outros *repositórios* vizinhos que estejam dentro da região de interesse. Dando continuidade ao exemplo acima, claramente

o *repositório* $R3$ pode responder apenas uma parte da consulta, ou seja, apenas os dados referentes ao agrupamento $C10$. Assim, $R3$ encaminha a consulta para o *repositório* $R4$, que pode responder com informações referentes aos agrupamentos $C6$, $C7$ e $C11$. Por fim, quando o processamento da consulta coletar todas as informações necessárias, o resultado da consulta retorna até o ponto de origem.

4. Avaliação do ASQPM

As avaliações do ASQPM e do IBIS ocorreram por meio de simulações no simulador de redes NS versão 2.35. No cenário empregado considerou-se uma região retangular de $1400m$ por $1000m$, nós idênticos e estáticos, distância entre os nós em torno de $90m$, com links simétricos, com raio de comunicação de $100m$ e protocolo MAC 802.11. Cada nó possui um GPS que informa sua posição sobre a região monitorada. No entanto podem-se assumir outras formas de localização, como a triangulação. As avaliações consideram também três cenários distintos para analisar a escalibilidade do ASQPM, compostos por 140, 560 e 1260 sensores espalhados sobre a área monitorada. No início das operações da rede, os nós executam o algoritmo para definição dos agrupamentos e dos nós *repositórios* de dados. Em seguida, o nó 28 inicia o processamento de uma consulta, a qual busca as médias das cinco leituras de todos os nós contidos dentro de um retângulo definido pelos vértices opostos $v_1(300, 50)$ e $v_2(950, 500)$, como ilustra a Figura 3. Os resultados mostrados nos gráficos são referentes aos dados coletados a partir do processamento desta consulta. Cada ponto plotado corresponde à média de 35 simulações, com intervalo de confiança de 95%. A Tabela 1 resume os principais parâmetros utilizados na simulação.

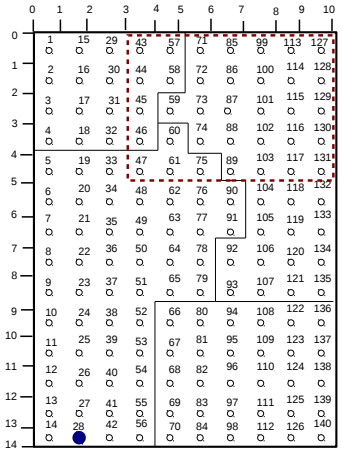


Figura 3. Cenário da simulação

Tabela 1. Parâmetros da simulação

Parâmetro	Cenário
Quantidade de sensores	$140, 560, 1260$
Fonte de dados	dados sintéticos
Área do ambiente	$1400m$ por $1000m$
Raio de comunicação do rádio	$100m$
$Threshold$ para agrupamento	$0.8 \leq \tau \leq 0.99$
$Threshold$ para *repositório*	$\tau >= 4$
Potência de transmissão	$0.051W$
Potência na recepção	$0.048W$

A avaliação dos experimentos realizou-se em duas etapas. O objetivo da primeira avaliação foi analisar o comportamento do modelo quanto à definição do número de *repositórios*, de líderes e de líderes isolados. Para cada simulação foram geradas novas leituras para os sensores distribuídos na área de monitoramento. Na segunda fase dos experimentos, o objetivo foi comparar o ASQPM com o IBIS [da Silva et al. 2011], analisando-se três métricas: **consumo de energia (Joules)**, **tempo gasto para processamento da requisição (segundos)** e **porcentual de erro nas leituras obtidas**. A energia consumida pela rede para transmitir um pacote (E_{PT}) é igual a energia para transmissão de um pacote (E_{TX}), mais a soma da energia consumida por cada um dos n vizinhos do

emissor para recebê-lo ($\sum_{x=1}^{n} E_{RE}$). Esse consumo pode ser modelado pela equação: $E_{(i \to j)PT} = E_{(i)TR} + \sum_{x=1}^{n} E_{(x)RE}$.

A geração dos dados de simulação levou em consideração a similaridade espacial dos nós na região monitorada. Assim, os nós sensores foram divididos em quatro regiões, como ilustrado pela Figura 3. Cada nó realizou a coleta de cinco leituras distintas, como temperatura e umidade, considerando-se que todas as leituras estão no intervalo entre 0 e 10. Inicialmente, foram criados quatro conjuntos de dados, sendo cada conjunto composto de 5 valores, gerados de forma aleatória. Cada conjunto foi associado a uma região e seus valores correspondem às sementes a partir das quais as leituras de cada nó sensor da região foram geradas. As leituras dos nós de uma mesma região são valores com variação aleatória de no máximo um porcentual V a partir do valor semente. Assim, a diferença na leitura dos nós de uma mesma região é no máximo $V\%$.

4.1. Formação de Agrupamentos e Repositórios

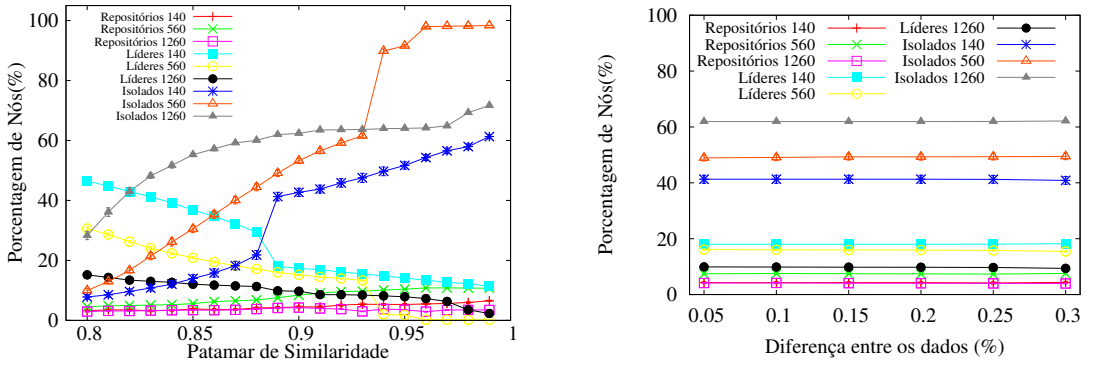
Nestes experimentos, o objetivo foi determinar a porcentagem de *repositórios*, líderes e líderes isolados gerados pelo ASQPM, em relação ao número de nós da rede. No gráfico da Figura 4(a), o eixo x representa o patamar de similaridade usado para definir os líderes (*threshold* do Algoritmo 1), o qual variou de 0.8 a 0.99, V foi fixado em 10% e o patamar para escolha de *repositórios* (*threshold* do Algoritmo 2) em todas as simulações foi 4.

Verifica-se que a porcentagem de *repositórios* praticamente não se altera. A definição de um *repositório* leva em consideração o número de líderes que um dado sensor é capaz de escutar. Por isso, a variação da similaridade praticamente não influencia a porcentagem de *repositórios* na rede. Em geral, a porcentagem máxima de nós selecionados como *repositórios* ficou abaixo de 10% em todos os cenários avaliados. Este resultado mostra que a estratégia do modelo de concentrar leituras em *repositórios* ao invés de mantê-los apenas nos líderes dos agrupamentos foi acertada. Como o processamento das consultas requer apenas acessos aos *repositórios*, o número de sensores contactados é pequeno, mesmo que as consultas espaciais cubram regiões geográficas extensas. O gráfico também mostra que, com o aumento do patamar de similaridade para definição de líderes, o número de líderes isolados cresce e de líderes com nós subordinados diminui. Isso ocorre em consequência da diminuição do número de nós com similaridade acima do patamar, o que gera mais líderes isolados. Além disso, o número de *repositórios* necessários para armazenar os dados de agrupamentos vizinhos também tende a ser menor quando há mais agrupamentos com nós subordinados do que agrupamentos com líderes isolados. Isso ocorre porque nesse caso o número total de agrupamentos formados tende a ser menor. Em cenários reais, nos quais há maior similaridade entre sensores vizinhos, acredita-se que a quantidade de líderes isolados seja ainda menor.

No gráfico da Figura 4(b), o eixo X representa a porcentagem de variação máxima V dos dados gerados como entrada para a simulação. Verifica-se que esse parâmetro não altera consideravelmente a formação de agrupamentos e a definição dos *repositórios*. Nestes experimentos, considerou-se um patamar de 0.88 de similaridade para os líderes.

Ainda nesse contexto, é importante analisar a quantidade de memória necessária para o funcionamento do ASQPM. Para isso, analisaremos somente o pior caso, a memória necessária para um *repositório*. Tal nó precisa armazenar sua posição geográfica (2 bytes), sua MBR (4 bytes), suas leituras (L * TL bytes, onde L é o número de leituras e TL

o espaço ocupado por uma leitura), a MBR e as leituras de cada um dos NCH líderes associados a ele e a posição geográfica e a MBR de cada um dos NR repositório conhecidos. Logo, o custo de armazenamento de um repositório pode ser calculado pela seguinte equação: $(6+(L*TL))+NCH*(4+L*TL)+(2+4)*NR$. No pior caso em nossos experimentos, ao considerar que um repositório sabe informações de todos os outros repositórios da rede e pela análise do gráfico ilustrado na Figura 4(a), temos $NR = 53$. Em análises dos logs dos experimentos, verificou-se que e o maior número de líderes associados a um repositório é $NCH = 54$. Logo, considerando $TL = 1$ byte, o custo máximo de armazenamento de um repositório foi de 834 bytes, menor que os 10 KB de memória RAM da maioria dos nós sensores descritos na literatura.

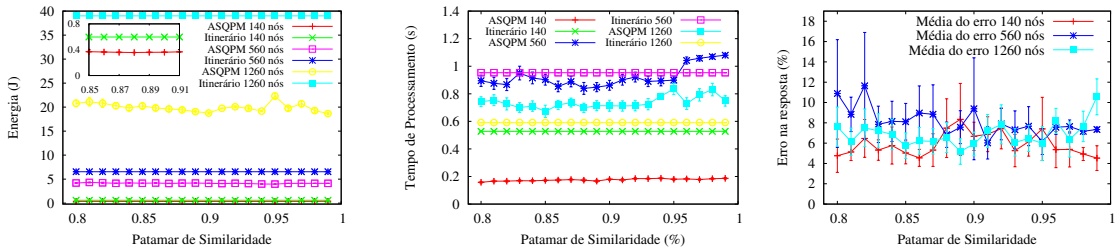


(a) Similaridade na geração de agrupamentos (b) Impacto do grau de similiaridade das leituras

Figura 4. Quantidade de líderes/agrupamentos e repositórios

4.2. Processamento de Consultas ASQPM e IBIS

Nesta seção, o processamento de consultas do ASQPM é comparado com o IBIS, o qual cria um itinerário dentro da região de coleta de forma que todos os sensores contidos nesta região são vizinhos do itinerário ou fazem parte dele. É importante salientar que o processamento de requisições do IBIS se altera somente com alterações na topologia da rede. Como nos experimentos aqui apresentados, a topologia da rede mantém-se inalterada, o comportamento do IBIS foi constante em todos os cenários analisados. Porém, dentre os trabalhos até então encontrados na literatura, tal algoritmo é o que apresenta o melhor desempenho no cenário utilizado na avaliação.



(a) Consumo de Energia (b) Tempo de Resposta (c) Perc. de Erro do Resultado

Figura 5. Comparação entre ASQPM e IBIS

O gráfico da Figura 5(a) mostra que o ASQPM consome menos energia que o IBIS, porque as requisições não precisam ser disseminadas para todos os nós na região de

consulta, basta elas alcançarem os *repositórios* que contém dados dessa região para se obter a resposta. Em relação ao tempo de resposta, apresentado pelo gráfico 5(b), para 140 nós o ASQPM é mais eficiente porque o IBIS cria vários *delays* durante a disseminação da consulta pelo itinerário. No cenário de 560 nós, o desempenho do ASQPM mostra-se inferior que o IBIS somente quando a similaridade ultrapassa 0.95, pois a partir deste limiar o ASQPM apresenta o maior número absoluto de *repositórios* de dados dentre os cenários analisados. No entanto, no cenário com 1260 nós, o ASQPM apresenta pior desempenho que o Itinerário. Isto ocorre pois o Itinerário tenta encaminhar a consulta sempre para o vizinho que estiver mais próximo da região de interesse. Dado que no cenário de 1260 cada nó apresenta vizinhos mais próximos dessa região, o encaminhamento e a transmissão da resposta até o nó que iniciou a requisição são realizados de maneira mais rápida.

De acordo com o gráfico 5(c), o ASQPM apresenta uma margem de erro enquanto que o IBIS retorna o resultado exato, uma vez que as leituras são obtidas diretamente dos dispositivos sensores da região de interesse. Os valores apresentados mostram a porcentagem de diferença entre a média das leituras de todos os sensores contidos na região de coleta e a média obtida com o processamento da requisição. Verifica-se que o erro do ASQPM varia em torno de 6% no cenário composto por 140 e 560 sensores, e relativamente um pouco maior para o cenário composto por 1260 sensores. Esse erro vem do fato que as leituras feitas por um líder representam a leitura do seu agrupamento. No entanto, é preciso levar em consideração que a média do erro é calculada a partir das leituras de 5 atributos de sensoriamento, o que geralmente não ocorre nos modelos tradicionais onde a média é calculada a partir de um único atributo do ambiente.

Além das métricas apresentadas, torna-se importante analisar o **custo energético do processo de definição de líderes e *repositórios***. O gasto energético do ASQPM ficou em torno de 0.35 Joules. Por outro lado, o consumo do IBIS foi menor, cerca de 0.054 Joules, pois necessita apenas de uma inundação na rede. Entretanto, essa diferença é compensada com o processamento de mais requisições. Como o ASQPM consome menos energia para processar, o consumo da rede durante todo seu período de operação tende a ser menor. Esses dados não são mostrados nos gráficos porque não ocorreram grandes variações dos valores acima mencionados. Estes resultados como um todo mostram que o ASQPM é um modelo que atende os requisitos de **escalabilidade** e **autonomia** para RSSFs urbanas. Mas, é preciso analisar e estender o trabalho com formas alternativas para reduzir o erro relativo das consultas. Observa-se que o ASQPM determina a similaridade entre sensores, considerando um *conjunto* de métricas, e não apenas um tipo de leitura.

5. Conclusão

Este trabalho apresentou um modelo para processamento de requisições espaciais em redes urbanas, chamado de ASQPM. O modelo foi inspirado em características normalmente encontradas em aplicações de sensoriamento urbano, como a alta densidade de sensores na rede para monitorar regiões extensas e a correlação espacial presente no dado sensoriado. Neste modelo, os dados de sensoriamento são distribuídos em estruturas denominadas de *repositórios*. Simulações mostraram que a estratégia adotada pelo ASQPM reduz significativamente o fluxo de mensagens durante o processamento de consultas. No entanto, o ASQPM mostrou uma taxa de erro pequena que deve ser levada em conta de acordo com os critérios da aplicação executada na rede. Como trabalho futuro, pretende-se analisar o impacto da dinâmica temporal do dado sensoriado.

Referências

Can, Z. and Demirbas, M. (2013). A survey on in-network querying and tracking services for wireless sensor networks. *Ad Hoc Networks*, 11(1):596–610.

Cheng, B., Xu, Z., Chen, C., and Guan, X. (2011). Spatial correlated data collection in wireless sensor networks with multiple sinks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 578–583. IEEE.

Coman, A., Sander, J., and Nascimento, M. A. (2007). Adaptive processing of historical spatial range queries in peer-to-peer sensor networks. *Distributed and Parallel Databases*, 22(2-3):133–163.

da Silva, R. I., Macedo, D. F., and Nogueira, J. M. S. (2011). Contornos irregulares no processamento de requisições espaciais para redes de sensores sem fio. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Da Silva, R. I., Macedo, D. F., and Nogueira, J. M. S. (2014). Spatial query processing in wireless sensor networks–a survey. *Information Fusion*, 15:32–43.

Furlaneto, S. S., Dos Santos, A., and Hara, C. S. (2012). An efficient data acquisition model for urban sensor networks. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 113–120. IEEE.

Hung, C.-C., Peng, W.-C., and Lee, W.-C. (2012). Energy-aware set-covering approaches for approximate data collection in wireless sensor networks. *Knowledge and Data Engineering, IEEE Transactions on*, 24(11):1993–2007.

Karp, B. and Kung, H.-T. (2000). Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM.

Le, T. D., Pham, N. D., and Choo, H. (2008). Towards a distributed clustering scheme based on spatial correlation in wsns. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC'08. International*, pages 529–534. IEEE.

Ma, Y., Guo, Y., Tian, X., and Ghanem, M. (2011). Distributed clustering-based aggregation algorithm for spatial correlated sensor networks. *Sensors Journal, IEEE*, 11(3):641–648.

Muller, C. L., Chapman, L., Grimmond, C., Young, D. T., and Cai, X. (2013). Sensors and the city: a review of urban meteorological networks. *International Journal of Climatology*, 33(7):1585–1600.

Resch, B., Mittlboeck, M., Girardin, F., Britter, R., and Ratti, C. (2009). Live geography–embedded sensing for standarised urban environmental monitoring.

Thepvilojanapong, N., Ono, T., and Tobe, Y. (2010). A deployment of fine-grained sensor network and empirical analysis of urban temperature. *Sensors*, 10(3):2217–2241.

Vuran, M. C., Akan, Ö. B., and Akyildiz, I. F. (2004). Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259.

Xie, L., Lu, S., Cao, Y., and Chen, D. (2014). Towards energy-efficient storage placement in large scale sensor networks. *Frontiers of Computer Science*, pages 1–17.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer networks*, 52(12):2292–2330.

Yoon, S. and Shahabi, C. (2007). The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 3(1):3.

Yu, Z., Xiao, B., and Zhou, S. (2010). Achieving optimal data storage position in wireless sensor networks. *Computer Communications*, 33(1):92–102.

MobDatU: Um Novo Modelo de Previsão de Mobilidade Humana para Dados Heterogêneos*

Lucas Maia Silveira[1], Jussara M. Almeida[1], Humberto Marques-Neto[2], Artur Ziviani[3]

[1]Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, MG – Brasil

[2]Pontifícia Universidade Católica de Minas Gerais (PUC MINAS)
Belo Horizonte, MG – Brasil

[3]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis, RJ - Brasil

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

Resumo. *Atualmente existem diversos modelos de mobilidade humana que visam descrever ou prever o comportamento humano em uma região durante um período de tempo. Entretanto, a maioria desses modelos foram avaliados utilizando dados de uma única fonte, por exemplo, dados de chamadas de telefonia móvel ou dados de GPS obtidos a partir de aplicações Web georreferenciadas. Portanto, a robustez destes modelos a diferentes tipos de dados é ainda desconhecida. Neste artigo, é proposto um novo modelo de previsão de mobilidade humana, chamado MobDatU, que foi projetado para utilizar tanto dados de telefonia móvel quanto dados de aplicativos georreferenciados (isolada e conjuntamente). O MobDatU, assim como dois modelos estado-da-arte, a saber SMOOTH e Leap Graph, são avaliados considerando diversos cenários com dados de fonte única e de múltiplas fontes. Os experimentos indicam que MobDatU atinge resultados melhores ou pelo menos comparáveis ao melhor modelo estado-da-arte em todos os cenários, diferentemente dos modelos alternativos cujo desempenho é bem mais sensível ao tipo de dado utilizado.*

Abstract. *Several previous mobility models aim at describing or predicting human behavior in a particular region during a certain period of time. Nevertheless, most of those models have been evaluated using data from a single source, such as data from mobile calls or GPS data obtained from Web applications. Thus, the effectiveness of such models when using different types of data remains unknown. This paper proposes a new model to predict human mobility, called MobDatU, which was designed to use data from mobile calls and data from georeferenced applications (in an isolated or combined way). MobDatU as well as two state-of-the-art models, namely SMOOTH and Leap Graph, are evaluated considering various scenarios with single data source and multiple data sources. The experiments indicate that MobDatU always produces results that are better than or at least comparable to the best baseline in all scenarios, unlike the previous models whose performance is very dependent on the particular type of data used.*

*Esta pesquisa é financiada pelo INCTWeb (MCT/CNPq 573871/2008-6), CNPq, CAPES, FAPEMIG e FAPERJ.

1. Introdução

O entendimento sobre a mobilidade humana pode ajudar governos e empresas a prever e planejar ações para melhorar a qualidade de vida das pessoas de uma determinada região. Tais previsões e planejamentos de ações podem ser feitas a partir de modelos de previsão de mobilidade, que por sua vez podem ser desenvolvidos a partir da análise de dados sobre deslocamentos de pessoas. Esses dados podem ser provenientes de diferentes fontes, incluindo telefonia móvel [Candia et al. 2008] e aplicativos georreferenciados [Noulas et al. 2012a], como o Twitter.

Embora estudos recentes mostrem que a mobilidade humana em áreas urbanas possa ser bem previsível considerando rotinas diárias [Song et al. 2010], a vasta maioria dos modelos de previsão de mobilidade disponíveis na literatura [Munjal et al. 2011, Dong et al. 2013, Lee et al. 2012, Allamanis et al. 2012] foi proposta ou pelo menos avaliada considerando uma única fonte de dados, tais como dados de chamadas de telefonia móvel ou dados coletados de aplicações Web georreferenciadas (*e.g.*, Twitter). A robustez de tais modelos a dados de fontes diferentes é ainda desconhecida. Mais ainda, o uso combinado de dados de diferentes fontes pode levar a previsões mais precisas ou à cobertura de um maior volume da população. Entretanto, a adequação e a eficácia dos modelos existentes a dados de múltiplas fontes também ainda são desconhecidas.

Neste contexto, este trabalho apresenta um novo modelo de previsão de mobilidade humana, chamado MobDatU, que foi projetado para explorar dados de fontes heterogêneas, especificamente dados de telefonia móvel e dados de aplicações georreferenciadas. O MobDatU utiliza alguns princípios abordados em dois outros modelos considerados estado-da-arte, o Leap Graph [Dong et al. 2013] e o SMOOTH [Munjal et al. 2011]. Enquanto o Leap Graph foi avaliado com dados de telefonia móvel, tendo conseguido um taxa de acerto de 84% no estudo original, o SMOOTH foi anteriormente avaliado somente com dados de aplicativos georreferenciados.

Neste artigo, nós comparamos a eficácia da previsão dos três modelos em vários cenários consistindo de: (i) dados homogêneos provenientes de fonte única, sejam logs de chamadas de telefonia móvel, sejam dados de GPS coletados da aplicação Twitter; e (ii) dados heterogêneos a partir da combinação de ambas as fontes. Os nossos resultados experimentais indicam que o novo modelo MobDatU alcança as maiores taxas de acerto em todos os cenários avaliados. Já o SMOOTH atinge resultados comparáveis ao do nosso modelo quando dados de GPS são utilizados, mas tem desempenho muito inferior nos outros cenários. O Leap Graph, por sua vez, obtém resultados similares ao do MobDatU para os cenários com dados de telefonia móvel (para os quais ele foi projetado) e também quando os dados das duas fontes são combinados. Entretanto, seu desempenho quando apenas dados de GPS são utilizados é inferior ao do SMOOTH e ao do MobDatU.

Portanto, as contribuições deste artigo são: (i) a avaliação de dois modelos de previsão de mobilidade estado-da-arte, SMOOTH e Leap Graph, em diferentes cenários com dados homogêneos e heterogêneos; e (ii) a proposição de um novo modelo, MobDatU, que produz resultados melhores ou pelo menos comparáveis ao melhor modelo tanto para os cenários de dados de telefonia móvel, quanto para os de Twitter.

O resto do artigo está organizado como segue. A Seção 2 discute a literatura sobre modelos de mobilidade humana. A Seção 3 formalmente apresenta a tarefa de previsão de

mobilidade humana, explica o funcionamento dos dois modelos de referência e apresenta o novo modelo MobDatU. A metodologia de avaliação adotada, incluindo as coleções de dados utilizadas nos experimentos, é descrita na Seção 4, enquanto os resultados são discutidos na Seção 5. Conclusões e trabalhos futuros são apresentados na Seção 6.

2. Trabalhos Relacionados

A literatura contém vários trabalhos que propõem modelos de previsão de mobilidade humana. Esses modelos utilizam diferentes tipos de dados e exploram diversas estratégias para tentar predizer com maior precisão a movimentação das pessoas (*i.e.*, usuários) [Bui et al. 2014]. Alguns destes modelos buscam prever as trajetórias das pessoas utilizando dados de GPS [Zheng and Xie 2011] ou dados de telefonia móvel [Dong et al. 2013]. Outros modelos realizam a previsão a partir de distribuições estatísticas que capturam padrões específicos identificados nos dados, tais como a distribuição da distância percorrida por um usuário [Lee et al. 2009, Rhee et al. 2008].

De forma similar, outros trabalhos, como [Jeung et al. 2008] e [Noulas et al. 2012b], exploram padrões diversos observados nos dados, tais como os lugares visitados pelas pessoas, a frequência de visitação e a regularidade dos padrões de locomoção de um usuário. Em [Noulas et al. 2012a], os autores exploram os locais visitados bem como as distâncias de locomoção de um usuário dentro de uma mesma região para prever sua localização em um momento futuro. Já em [Cho et al. 2011], os autores abordam a previsão de mobilidade considerando tanto trajetórias longas quanto trajetórias curtas.

Utilizando dados de GPS coletados de aplicações de redes sociais georreferenciadas, os modelos propostos em [Musolesi and Mascolo 2007] e [Nguyen and Szymanski 2012] exploram não somente os padrões de distância percorrida e locais visitados, mas também as relações de amizade entre os usuários (inferidas a partir dos dados coletados). Ou seja, os modelos pressupõem que uma pessoa terá maior chance de ir para um local se um grande número de seus amigos frequentarem esse local.

A previsão da mobilidade humana pode auxiliar na prevenção de doenças e desastres bem como no planejamento urbano. Por exemplo, os trabalhos de [González et al. 2008, Balcan et al. 2009] mostram que o conhecimento sobre a mobilidade das pessoas em uma determinada região pode auxiliar na tomada de decisões para melhor e mais rapidamente evitar a disseminação de uma doença, amenizar os danos causados por desastres e até mesmo evitar engarrafamentos durante as horas de pico. Além disso, tal conhecimento também pode auxiliar as operadoras a aperfeiçoarem seus serviços de telefonia móvel [Xavier et al. 2012].

Entretanto, os modelos de mobilidade humana disponíveis na literatura foram propostos ou avaliados considerando apenas uma fonte única de dados, sejam chamadas de telefonia móvel [Dong et al. 2013, Candia et al. 2008, Balcan et al. 2009] sejam dados de GPS coletados de aplicações georreferenciadas [Lee et al. 2009, Rhee et al. 2008, Munjal et al. 2011, Musolesi and Mascolo 2007, Noulas et al. 2012a, Nguyen and Szymanski 2012, Cho et al. 2011]. A robustez desses modelos a dados de fontes diversas, seja isolada ou conjuntamente, é o foco deste trabalho. Para tanto são selecionados dois modelos de referência, o SMOOTH [Munjal et al. 2011] e o Leap Graph [Dong et al. 2013]: enquanto o primeiro foi avaliado anteriormente somente a

partir de dados de GPS, o segundo foi avaliado para dados de telefonia móvel. Mais ainda, é proposto um novo modelo que herda princípios dos dois modelos de referência, mas é projetado para explorar dados de fontes heterogêneas. Os dois modelos de referência, bem como o novo modelo proposto, chamado MobDatU, são descritos na próxima seção.

3. Modelos de Mobilidade

Esta seção primeiramente apresenta formalmente a tarefa de previsão de mobilidade humana abordada neste artigo (Seção 3.1). A seguir, ela descreve brevemente o funcionamento dos dois modelos de referência, SMOOTH e Leap Graph (Seção 3.2) e introduz o novo modelo de previsão proposto, o MobDatU (Seção 3.3).

3.1. Previsão de Mobilidade Humana

A tarefa de previsão de mobilidade abordada neste artigo pode ser definida como segue. Sejam uma área alvo R, consistindo de um conjunto de regiões $r_i \in R$, um conjunto de usuários $u_i \in U$ e um intervalo de tempo $t \in \{0..T\}$ minutos. Sejam ainda um conjunto de treino $\mathcal{D}$ e um conjunto de teste $\mathcal{T}$, consistindo de tuplas $< u_i, r_i, t >$ que indicam que u_i estava na região r_i em t. Deseja-se construir um modelo para prever em que região r_i um dado usuário u_i estará em um dado momento t utilizando apenas os dados de treino $\mathcal{D}$. Deseja-se ainda avaliar o modelo utilizando os dados de teste, ou seja, utilizar o modelo desenvolvido para prever a localização (*i.e.*, região) r_i para cada tupla $< u_i, ?, t >$ no conjunto de teste $\mathcal{T}$.

Note que a definição das regiões r_i pode ser feita de diversas maneiras. Por exemplo, cada região pode ser definida por um ponto central x_i, y_i (e.g, as coordenadas de uma estação rádio-base para o caso de dados de telefonia móvel) e um raio d. Alternativamente, cada região pode representar uma área quadrada com centro x_i, y_i e lado d (área total d^2). A definição das regiões adotada por cada modelo é explicitada nas seções seguintes. Note também que, neste trabalho, o intervalo de tempo total T é discretizado em janelas consecutivas Δt para fins de computação dos deslocamentos dos usuários. Em outras palavras, a localização de um usuário é predita considerando a granularidade de tempo de Δt minutos.

A divisão dos dados disponíveis em treino e teste também pode ser feita de forma diversa, dependendo da disponibilidade dos dados. Entretanto, tal divisão deve respeitar restrições temporais, ou seja, os dados de treino devem preceder os dados de teste (*i.e.*, $\forall <*, *, t_i> \in \mathcal{D}$ and $\forall <$*,*,$t_j> \in \mathcal{T}$, $t_i < t_j$). Mais ainda, considerando que os padrões de mobilidade humana variam ao longo do dia ou mesmo em dias diferentes (*e.g.*, dias de semana e fim de semana) [Xavier et al. 2012], é desejado que os dados de treino tenham sido obtidos em períodos comparáveis com aqueles do conjunto de teste. Por exemplo, se deseja prever a localização de usuários entre 8 e 9 da manhã, deve-se utilizar dados coletados do mesmo período em dias anteriores, ou dados coletados em um período imediatamente anterior. A definição dos conjuntos de treino e teste em nossos experimentos será discutida na Seção 4.1.

3.2. Modelos de Referência

Esta seção descreve os principais componentes dos dois modelos de referência adotados neste trabalho, o SMOOTH [Munjal et al. 2011] e o Leap Graph

[Dong et al. 2013]. Em nossos experimentos, nós utilizamos as implementações de ambos os modelos disponibilizadas pelos autores.[1]

3.2.1. SMOOTH

O modelo SMOOTH, proposto por [Munjal et al. 2011], captura a locomoção de um grupo de usuários U em uma área bidimensional simulada, composta por um conjunto de regiões de interesse. Cada região r_i é definida por coordenadas x_i, y_i dentro da área simulada e probabilidades p_i de um usuário se deslocar para cada uma delas. A probabilidade p_i portanto captura a popularidade da região r_i, isto é, o número esperado de pessoas que visitam r_i.

A ideia básica do SMOOTH é simular a locomoção dos usuários U utilizando cada probabilidade p_i de forma interativa, onde cada passo representa uma janela de tempo Δt minutos. Em cada passo, o modelo simula a locomoção de cada usuário $u_i \in U$ a partir de duas distribuições específicas extraídas do conjunto de treino: a distribuição das distâncias percorridas f_{dist} e a distribuição dos tempos de pausa f_{pausa}. Como em [Munjal et al. 2011], observamos que tais distribuições seguem leis de potência para todos os cenários simulados. Tais distribuições são caracterizadas pelo parâmetro α e por valores mínimo (min) e máximo (max)[2].

Em cada passo, para cada usuário que não está em pausa (explicado posteriormente), primeiramente é computada a direção de deslocamento em função da sua localização atual e das probabilidades associadas a cada região $r_i \in R$. Em seguida, é selecionada aleatoriamente uma distância de deslocamento utilizando f_{dist}, e o deslocamento é simulado. Por fim, é escolhido aleatoriamente um tempo de pausa utilizando f_{pausa}. O usuário permanecerá neste local durante o intervalo de tempo f_{pausa} selecionado.

Conforme proposto em [Munjal et al. 2011], a cada usuário é associado um raio de cobertura d que definirá a região na qual o usuário se encontra em cada passo de execução do modelo. Quando o usuário para em um ponto (x_i, y_i), é verificado se ele se encontra a uma distância d de pelo menos uma região (ou seja, se ele cobre uma região). O usuário é associado à região mais próxima coberta por ele. Caso ele não cubra nenhuma região, é introduzida uma nova região r_i em R definida pela sua localização atual. Ao final de cada passo, as probabilidades associadas a cada região (inclusive as novas) são recomputadas.

Logo, o treinamento do modelo consiste em extrair as distribuições f_{dist} e f_{pausa} bem como as regiões $r_i \in R$ do conjunto $\mathcal{D}$. Todas as regiões que aparecem no conjunto de treinamento são inicialmente introduzidas no conjunto R com as probabilidades correspondentes. Estas probabilidades são utilizadas para determinar as localizações iniciais dos usuários para a fase de simulação dos deslocamentos, durante a qual são aprendidas novas regiões visitadas e suas probabilidades.

Durante a fase de teste, os deslocamentos dos usuários do conjunto $\mathcal{T}$ são simulados utilizando o modelo aprendido, mantendo o conjunto R fixo e considerando

[1] toilers.mines.edu e www.cs.utexas.edu/ wdong86/.

[2] Distribuição Acumulada Complementar dada por $f(x) = \frac{(x*max^{\alpha} - x*min^{\alpha} - max^{\alpha})}{(x*max^{\alpha})*(x*min^{\alpha})}^{-\alpha}$.

a localização inicial de cada usuário dada pela sua primeira aparição em $\mathcal{T}$. As regiões visitadas pelos usuários durante a simulação são comparadas com os demais dados do conjunto $\mathcal{T}$ para avaliar a precisão do modelo.

3.2.2. Leap Graph

Em [Dong et al. 2013], os autores investigaram como utilizar dados de telefonia móvel para prever a mobilidade de usuários. Estes dados tipicamente correspondem a um conjunto de chamadas telefônicas. A cada chamada estão associados um identificador único do usuário, os instantes de início e fim da chamada, bem como as coordenadas (latitude e longitude) das antenas onde a chamada foi iniciada e finalizada e as identificações dos setores utilizados nestas antenas[3].

No modelo proposto em [Dong et al. 2013], chamado Leap Graph, cada região $r_i \in R$ corresponde a uma antena, sendo definida pelas suas coordenadas e por um raio d que representa sua área de cobertura. Assim, uma chamada do usuário u_i associada à antena correspondente à região r_i em um instante t indica a presença de u_i naquela região na janela de tempo que inclui t. O modelo tenta inferir os deslocamentos de cada usuário a partir de um grafo que captura a trajetória do usuário entre as regiões de R.

A fase de treinamento, portanto consiste primeiramente em criar um grafo de trajetórias para cada usuário a partir dos dados em $\mathcal{D}$. No grafo G_i criado para o usuário u_i, cada vértice corresponde a uma região. Uma aresta entre r_i e r_j é adicionada toda vez que: (i) u_i fez uma chamada que foi iniciada em r_i e finalizada em r_j; ou (ii) u_i fez duas chamadas consecutivas, a primeira em r_i e a segunda em r_j.

Como proposto originalmente, o Leap Graph objetiva prever a próxima região em que um usuário estará dado a sua localização atual. Portanto, ele não considera a dimensão tempo e explora apenas as transições entre antenas feitas por cada usuário. Para torná-lo comparável ao SMOOTH e ao MobDatU (que consideram o tempo) e aplicá-lo à tarefa de predição alvo, nós adicionamos laços (arestas da região r_i para ela mesma) para capturar os períodos entre chamadas consecutivas de um mesmo usuário, quando sua localização é desconhecida. Durante tais períodos, foi assumido que o usuário permaneceu a metade do tempo em uma região e a metade seguinte na outra. Ou seja, dadas duas chamadas consecutivas nas regiões r_i e r_j nos tempos t_1 e t_2 e considerando a discretização do tempo em janelas de duração Δt, foi computado o número w de janelas entre t_1 e t_2. Foram então adicionados laços em r_i e r_j, cada um com peso $w/2$.

Os grafos criados são então combinados em um grafo único ponderado G que representa os deslocamentos da população de usuários em $\mathcal{D}$. Para tal, os grafos de usuários G_i são ordenados pelo instante da primeira chamada de cada usuário em $\mathcal{D}$ e processados conforme esta ordem. As arestas de todos os grafos são combinadas em G, sendo que o peso de uma aresta corresponde ao número de grafos de usuários em que ela aparece. Porém, para trajetórias cobrindo n ou mais arestas que aparecem em múltiplos grafos, são considerados apenas a trajetória e os trechos que a sucedem no primeiro grafo processado. Em outras palavras, suponha que o grafo G_1 contenha a trajetória

[3]Cada antena é dividida em 3 setores de 120^o, cada um responsável por cerca de um terço da área de cobertura da antena.

$\{r_1, r_2, r_3, r_4\}$ e que grafo G_2 contenha a trajetória $\{r_1, r_2, r_3, r_5\}$. Para $n = 2$, o grafo G conterá as arestas $\{r_1, r_2, r_3, r_4\}$, todas com peso 1, já que a trajetória $\{r_1, r_2, r_3\}$ ($n = 2$ arestas) aparece nos dois grafos. Conforme os autores, esta medida é tomada para evitar contabilizar duplamente as mesmas trajetórias. Ao final da combinação, os pesos de todas as arestas são normalizadas de forma que os pesos de todas as arestas de saída de cada vértice r_i totalizem 1.

A aplicação do modelo Leap Graph, durante a fase de teste, consiste em simular o grafo G produzido durante o treinamento como uma cadeia de Markov. Para cada usuário, a sua posição inicial é extraída da sua primeira chamada no conjunto $\mathcal{T}$ e que corresponde a um estado da cadeia. A cadeia é então simulada para inferir a posição do usuário em sucessivos passos. Em nossos experimentos, utilizamos $n = 2$ pois esta escolha levou aos melhores resultados em [Dong et al. 2013].

3.3. MobDatU: Um Novo Modelo de Predição de Mobilidade Humana

O novo modelo MobDatU tem como objetivo explorar dados de fontes heterogêneas para capturar a movimentação dos usuários entre as regiões de R. A área total simulada pelo modelo é divida em regiões quadrangulares não sobrepostas r_i (como em um *grid*). Cada região r_i é definida por um centro x_i, y_i e um lado d[4].

O MobDatU herda alguns aspectos dos dois modelos de referência. Por exemplo, assim como no SMOOTH, a cada região r_i é associada uma medida de popularidade, que, no MobDatU, representa o número de usuários que visitou r_i no conjunto de treino $\mathcal{D}$ (podendo ser 0). De forma similar ao Leap Graph (e diferentemente do SMOOTH), o MobDatU simula a movimentação dos usuários entre as regiões a partir de um grafo de transições. Entretanto, diferentemente do Leap Graph, a criação deste grafo não parte das trajetórias individuais de cada usuário e também não inclui o descarte de trajetórias com prefixo comum durante o processo de combinação dos grafos de usuários.

Ao invés disto, o MobDatU cria um grafo de transições onde o peso associado a uma aresta (r_i, r_j) representa o número de pessoas que fizeram a transição entre as duas regiões no conjunto $\mathcal{D}$. Este número pode ser inferido a partir tanto de dados de aplicativos georreferenciados quanto de dados de telefonia móvel. No segundo caso, assim como no Leap Graph, transições *self loop* são introduzidas para capturar os períodos entre chamadas sucessivas de um mesmo usuário.

A fase de treinamento do modelo consiste, portanto, em aprender o grafo de transições (incluindo os pesos das arestas) e as popularidades de cada região a partir do processamento do conjunto $\mathcal{D}$. Ao final, os pesos de todas as arestas são recomputados para capturar as popularidades de cada região destino. Em outras palavras, o peso da aresta (r_i, r_j) é multiplicado pela popularidade p_j. Os pesos das arestas saindo de cada vértice são então normalizados para refletir probabilidades de transição (ou seja, devem somar 1). Este é um ponto em que o MobDatU difere tanto do Leap Graph, que considera somente as probabilidades de transição, quanto do SMOOTH, que explora somente as popularidades de cada região. O MobDatU considera que ambos aspectos podem influenciar a trajetória de um usuário: se por um lado os usuários tendem a visitar locais específicos dependendo da sua localização atual (como mostrado em [Dong et al. 2013]),

[4]O modelo também foi executado utilizando regiões circulares de raio d, apresentando resultados similares aos reportados neste artigo.

Tabela 1. Coleções de Dados Utilizadas

	Chamadas		*Tweets*		Intervalo de Tempo
	# Chamadas	# Usuários	#*Tweets*	#Usuários	
Fortaleza - 29/06/14	7185	2372	13453	4236	14h - 21h
Recife - 29/06/14	13335	4923	13577	3981	14h - 21h
Belo Horizonte - 02/03/13	15630	9354	14332	4870	12h - 19h
Belo Horizonte - 11/09/13	14023	4532	15635	5103	17h - 23h
Rio de Janeiro - 29/06/14	5120	1132	14033	3643	14h - 21h
Rio de Janeiro - 13/07/14	5340	1038	15860	4572	14h - 21h

por outro, a popularidade de uma região também influencia a movimentação dos usuários [Munjal et al. 2011, Noulas et al. 2012a].

Durante a fase de teste, é simulada uma cadeia de Markov utilizando as probabilidades de cada transição, como no Leap Graph, e considerando a primeira posição de cada usuário como sendo sua posição inicial em $\mathcal{T}$.

4. Metodologia

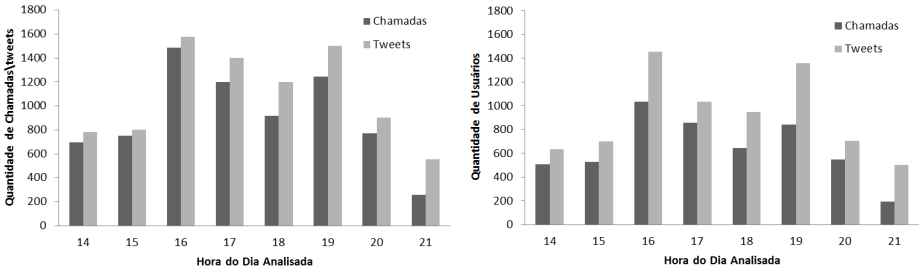
Nessa seção serão apresentadas as coleções de dados que foram utilizadas pelos modelos (Seção 4.1) e a metodologia de avaliação adotada (Seção 4.2).

4.1. Coleções de Dados

Para avaliar os modelos de predição foram utilizados dados de telefonia móvel e dados coletados do Twitter. Os dados de chamadas foram fornecidos por uma grande companhia de telefonia móvel brasileira e correspondem às chamadas realizadas em algumas cidades brasileiras durante intervalos de tempo pré-especificados. Já os dados do Twitter consistem em *tweets* georreferenciados. Eles foram coletados pela *Stream API* da aplicação [Twitter 2013] utilizando o filtro *location* que restringe a área de coleta para uma determinada região. A coleta pela API é realizada em tempo real. Logo, após a companhia de telefonia móvel informar os dados que seriam coletados, a coleta de *tweets* foi planejada para os mesmos períodos de tempo e locais. Os dados coletados são: o identificador do usuário que realiza a chamada/*tweet* e a posição geográfica da antena (em caso de chamada) ou do *tweet*. Como as duas coletas são independentes, não é possível identificar a mesma pessoa em coleções diferentes. Logo, os usuários nas duas coletas são tratados como diferentes.

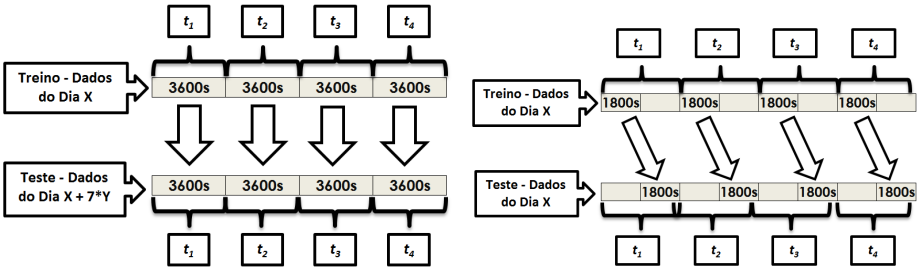
Os dados coletados foram filtrados para retirar os usuários que realizaram somente uma chamada ou publicaram somente um *tweet* em todo o período de tempo analisado. A Tabela 1 apresenta um sumário das coleções filtradas, informando local e período de cobertura, bem como números de chamadas, *tweets* e usuários. Note que o volume de *tweets* é similar em todas as coleções, enquanto o volume de chamadas é muito maior nas coleções de Recife e Belo Horizonte, sendo comparável ao volume de *tweets* coletados no mesmo período. Note também a variação na cobertura de usuários. Para Belo Horizonte (02/03/13), o número de usuários é duas vezes maior na coleção de chamadas. Já para as coleções do Rio de Janeiro, o número de usuários é muito maior na coleção de *tweets*. Estas diferenças podem ser explicadas por aspectos socioculturais e eventos específicos que ocorreram em cada cidade nos períodos monitorados.

A Figura 1 mostra os números de chamadas, *tweets* e usuários ao longo do tempo nas coleções do Rio de Janeiro (29/06/2014). Esta figura ilustra que o volume de dados



(a) #Chamadas/*Tweets* (b) #Usuários

Figura 1. Quantidade de Chamadas/*Tweets* e Usuários por Hora - Rio de Janeiro 29/06/14



(a) Utilizando Dois Dias (b) Utilizando Um Dia

Figura 2. Separação dos dados em Treino e Teste

disponível varia bastante segundo o horário de coleta, o que é esperado também para os padrões de movimentação das pessoas. Este resultado motiva a aplicação de modelos de predição específicos para diferentes períodos, conforme descrito na próxima seção.

4.2. Avaliação

Cada modelo de previsão foi avaliado em termos da taxa de acerto[5] das previsões em janelas de tempo Δt iguais a 5 minutos. Para tal, cada coleção foi subdividida em intervalos de uma hora. Dadas as variações nos volumes de dados observadas (Figura 1), optou-se por desenvolver um modelo de predição para cada hora de forma a melhor capturar os padrões de locomoção em diferentes períodos. Em seguida, os dados de cada subcoleção foram divididos entre treino $\mathcal{D}$ e teste $\mathcal{T}$, conforme discutido na Seção 3.1. Duas estratégias foram adotadas para fazer esta divisão, como descrito a seguir.

Para as coleções do Rio de Janeiro, como ambas cobrem o mesmo período no mesmo dia de semanas diferentes, a coleção do dia 29/06/14 foi utilizada como treino para aprender os modelos de cada hora. Estes modelos foram avaliados nos períodos correspondentes da base do dia 13/07/2014 (teste). Esta estratégia é mostrada na Figura 2(a). Para as demais coleções, como não temos acesso a dados de múltiplos dias cobrindo o mesmo período, foi adotada uma estratégia diferente. Cada hora no intervalo de tempo coberto pela coleção foi dividida em dois períodos de meia hora. O primeiro período foi utilizado para aprender o modelo (treino), e o segundo para avaliá-lo (teste). Esta estratégia é ilustrada na Figura 2(b).

[5]A taxa de acerto é calculada como a fração de tuplas $< u_i, r_i, t >$ para as quais a previsão foi correta durante a fase de teste

Para cada coleção, cada modelo de mobilidade foi avaliado usando, como conjuntos de treino e teste, somente as chamadas, somente os *tweets* e tanto chamadas quanto *tweets*. O objetivo do último cenário é avaliar o desempenho dos modelos quando configurados com dados heterogêneos. A melhor forma de combinar os dados das duas fontes não é óbvia já que modelos diferentes podem ter desempenhos relativos diferentes, dependendo do dado de entrada. Assim, nós consideramos e avaliamos duas estratégias:

- Associação de *tweets* a chamadas: cada *tweet* é associado à antena mais próxima de sua localização.
- Associação de chamadas a *tweets*: cada antena da coleção de chamadas é considerado um ponto na região de simulação.

Um aspecto importante da avaliação é a definição das regiões R de cada modelo. Para as chamadas, o Leap Graph e o SMOOTH consideram a localização de cada antena que aparece no conjunto $\mathcal{D}$ como uma região r_i. Porém no caso do SMOOTH, novas regiões podem surgir durante o treino, de acordo com a movimentação dos usuários. Para os *tweets*, o SMOOTH considera inicialmente cada localização distinta associada a um *tweet* em $\mathcal{D}$ como uma região[6], e novas regiões podem surgir durante o treino. Para o Leap Graph, os *tweets* foram agrupados em regiões circulares (com raio d) considerando suas localizações, e estas regiões foram inseridas em R. Para os dados combinados, foram adotadas as mesmas abordagens dependendo da estratégia de combinação feita. Já o MobDatU sempre divide a área total de cada cidade em regiões quadrangulares não sobrepostas (Seção 3.3), independentemente do tipo de dado.

Em todos os casos, consideramos a distância d que define cada região como 500 metros. Este valor foi escolhido para que os resultados dos vários cenários pudessem ser comparados[7], uma vez que, para os dados de telefonia móvel, a posição geográfica informada tem granularidade dada pelo raio de cobertura de uma antena, a saber 500 metros .

5. Experimentos

Nossos resultados são sumarizados na Tabela 2 que apresenta, para cada cenário e para cada coleção de dados, os valores de taxa de acerto mínimo, máximo e médio considerando todos os intervalos de tempo $\Delta t = 5$ minutos em todas as horas do período de cada coleção. No geral, nota-se que cada modelo de referência funciona melhor se configurado para as fontes de dados homogêneas no qual ele foi avaliado anteriormente: o Leap Graph consegue taxa de acertos melhores quando utiliza somente chamadas, e o SMOOTH consegue resultados melhores quando utiliza somente *tweets*.

Por exemplo, considerando a coleção de dados do Rio de Janeiro, nota-se que, quando os dados de chamadas são usados como entrada, o SMOOTH tem um desempenho médio degradado de 53% em relação ao Leap Graph. Já quando são usados os *tweets*, o Leap Graph possui um desempenho de 14% pior que o SMOOTH. Quanto aos cenários com dados heterogêneos, a taxa de acerto de cada modelo foi um pouco menor em relação

[6]Como descrito na Seção 3.2.1, cada região é definida no SMOOTH por um ponto, e a associação de um usuário a uma região é feita verificando um raio de cobertura d a partir do *usuário*.

[7]Note que as áreas das regiões do SMOOTH e do Leap Graph (πd^2) são maiores que as áreas das regiões definidas pelo MobDatU (d^2). Logo a nossa avaliação favorece os modelos de referência, uma vez que as predições são mais difíceis para regiões com áreas menores.

Tabela 2. Taxa de Acerto dos Modelos de Previsão de Mobilidade

Rio de Janeiro								
Modelo	Chamadas				Tweets			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	18%	55%	36,75%	0,13	34%	73%	52,12%	0,138
Leap Graph	68%	88%	78,13%	0,072	21%	60%	44,75%	0,129
MobDatU	70%	87%	79,88%	0,063	37%	77%	55%	0,143
Modelo	*Tweets* a Chamadas				Chamadas a *Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	45%	52%	48%	0,26	38%	77%	51,5%	0,13
Leap Graph	65%	84%	73,75%	0,062	25%	58%	41,63%	0,12
MobDatU	67%	85%	74,38%	0,059	40%	79%	53,12%	0,129
Belo Horizonte 02/03/2013								
Modelo	Chamadas				*Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	20%	48%	35,37%	0,104	34%	70%	51,25%	0,125
Leap Graph	68%	80%	73%	0,044	21%	50%	40,75%	0,097
MobDatU	67%	82%	75,12%	0,047	37%	77%	54,5%	0,144
Modelo	*Tweets* a Chamadas				Chamadas a *Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	20%	35%	28,37%	0,053	38%	73%	50,85%	0,117
Leap Graph	62%	77%	67,75%	0,054	30%	50%	36,75%	0,07
MobDatU	58%	78%	66,87%	0,071	37%	77%	52,37%	0,131
Belo Horizonte 11/09/2013								
Modelo	Chamadas				*Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	18%	40%	27,25%	0,08	38%	77%	53,25%	0,130
Leap Graph	67%	75%	72,25%	0,025	22%	50%	36,5%	0,092
MobDatU	70%	81%	73,75%	0,043	43%	78%	55,13%	0,128
Modelo	*Tweets* a Chamadas				Chamadas a *Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	18%	32%	25,75%	0,051	42%	78%	57,62	0,138
Leap Graph	67%	86%	76,13%	0,063	22%	67%	41,75%	0,152
MobDatU	67%	87%	75%	0,067	42%	80%	57,87	0,14
Recife								
Modelo	Chamadas				*Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	20%	35%	27,25%	0,053	42%	78%	43,12	0,139
Leap Graph	67%	86%	75,75	0,062	25%	67%	43,12%	0,139
MobDatU	67%	87%	75%	0,067	42%	80%	57,87%	0,140
Modelo	*Tweets* a Chamadas				Chamadas a *Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	19%	34%	26,13%	0,061	42%	77%	53,38%	0,121
Leap Graph	69%	75%	72,12%	0,023	22%	45%	35,13%	0,075
MobDatU	70%	77%	73%	0,031	43%	78%	54,63%	0,124
Fortaleza								
Modelo	Chamadas				*Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	18%	35%	26,25%	0,062	40%	77%	56,63%	0,138
Leap Graph	65%	84%	73,75%	0,063	25%	67%	42,75%	0,140
MobDatU	67%	85%	74,37%	0,059	42%	80%	57,87%	0,14
Modelo	*Tweets* a Chamadas				Chamadas a *Tweets*			
	Mínimo	Máximo	Média	Desvio Padrão	Mínimo	Máximo	Média	Desvio Padrão
SMOOTH	22%	37%	28,75%	0,053	42%	77%	52,12%	0,118
Leap Graph	62%	75%	66,87%	0,039	30%	50%	37,63%	0,078
MobDatU	56%	78%	67,37%	0,072	38%	79%	52,25%	0,129

ao seu melhor cenário. Ou seja, os modelos de referência se comportam um pouco pior em cenários com dados heterogêneos.

Em contrapartida, o MobDatU tem um desempenho comparável aos melhores resultados dos modelos de referência em todos os cenários. Para o cenário de chamadas, ele tem uma taxa de acerto ligeiramente maior que a do Leap Graph e muito superior (109%) à do SMOOTH. Já para o cenário de *tweets*, os resultados do Leap Graph são piores em 19% quando comparados aos do MobDatU, enquanto o SMOOTH consegue resultados bem próximos aos do nosso modelo. Comparando os dois cenários com dados homogêneos, nota-se que o MobDatU tem uma taxa de acerto bem maior no cenário de chamadas. A razão para este resultado é um maior número de regiões distintas presentes somente no conjunto de teste para os dados de *tweets*. Regiões que não aparecem no conjunto de treino terão popularidade e taxas de transição nulas no modelo e nunca serão previstas. Logo, regiões que só aparecem no teste necessariamente levam a previsões erradas do modelo[8].

Em relação ao cenário com dados heterogêneos, assim como os modelos de referência, o MobDatU apresenta um desempenho um pouco pior em relação aos cenários de dados homogêneos, mas ainda superior, em média, ao dos modelos de referência. Quanto às estratégias de combinação de dados, observa-se que cada modelo de referência teve melhor desempenho quando a estratégia adotada faz a associação tendo como alvo o tipo de dado para o qual o modelo tem melhor precisão (chamadas para o Leap Graph e para o MobDatU, *tweets* para o SMOOTH).

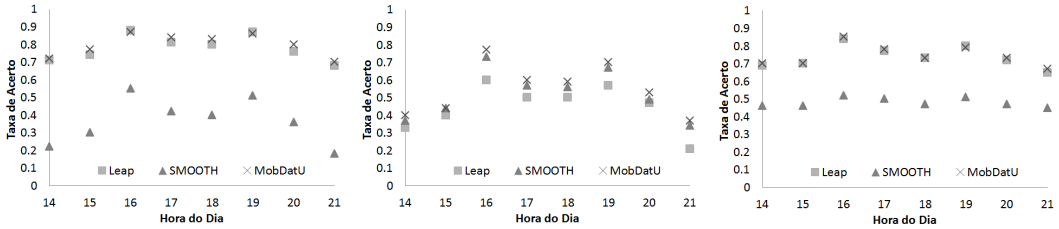
Os resultados obtidos para as outras coleções de dados apresentam um comportamento similar aos dados do Rio de Janeiro. Uma exceção ocorre na coleção de Belo Horizonte (11/09/2013), na qual o MobDatU teve resultados melhores nos cenários com dados heterogêneos. Vale ressaltar que o uso de dados heterogêneos viabiliza uma realização de previsões para um número de usuários (isto é cobertura de usuários) potencialmente muito maior[9], o que pode compensar eventuais perdas nas taxas de acerto quando comparadas às obtidas com dados homogêneos.

A Figura 3 mostra as taxas de acerto de cada modelo ao longo do período coberto pela coleção do Rio de Janeiro para os cenários com dados homogêneos e para um cenário com dado heterogêneo. Comparando esta figura com a Figura 1, nota-se que as maiores taxas de acertos foram obtidas em períodos com maior o volume de dados, o que era esperado. A figura também mostra que os ganhos do MobDatU sobre o melhor modelo de referência podem ser maiores que os mostrados na Tabela 2. Por exemplo, na Figura 3(b), para o intervalo de 16 horas, a taxa de acerto do MobDatU supera em 6% a do SMOOTH (melhor modelo de referência), enquanto na Figura 3(a), o MobDatU supera o Leap Graph em 5% para o período de 20 horas. Resultados semelhantes foram observados também para as outras coleções.

Em suma, o MobDatU mostrou se adequar bem a dados distintos, conseguindo desempenho comparável (e por vezes superior) ao do melhor modelo de referência,

[8]Note que este maior número de regiões novas no teste foi observada em todas coleções, a despeito das diferenças de volumes de dados de chamadas e *tweets* discutidos na Seção 4.1.

[9]Já que as coleções de chamadas e *tweets* não apresentam nenhum parâmetro que indicasse a existência de um mesmo usuário em ambas as fontes, foi considerado que os usuários dos dados de chamadas eram diferentes dos de *tweets* conforme descrito na Seção 4.1.



(a) Chamadas (b) *Tweets* (c) *Tweets* para Chamadas

Figura 3. Taxa de Acerto para Cenários 1, 2 e 3- Rio de Janeiro

em todos os cenários com dados homogêneos e heterogêneos. Em contrapartida, tanto SMOOTH quanto Leap Graph podem ter grande perda de desempenho, dependendo do tipo de dado de entrada. A superioridade do nosso modelo pode ser explicada por ele considerar tanto a popularidade das regiões quanto a frequência de transições entre as regiões. Ambos os aspectos são fatores importantes que influenciam como as pessoas se movimentam em uma cidade. Além disto, como estes aspectos podem ser capturados tanto pelas chamadas telefônicas realizadas quanto pelos *tweets* compartilhados, o MobDatU mostrou um bom funcionamento para ambos os tipos de dados.

6. Conclusão e Trabalhos Futuros

Neste artigo, foi proposto um novo modelo de previsão de mobilidade humana, o MobDatU. O modelo proposto, assim como dois modelos estado-da-arte, foram avaliados em diversos cenários com dados reais homogêneos e heterogêneos. Os experimentos indicaram que nenhum dos dois modelos de referência é superior em todos os cenários investigados, o que demonstra a sensibilidade dos mesmos aos dados disponíveis. Já o MobDatU se adequou bem a ambos tipos de dados, sendo pelo menos comparável (e por vezes superior) ao melhor modelo de referência em todos os cenários. Também foi mostrado que o volume de dados, tanto de chamadas telefônicas quanto de *tweets*, varia bastante durante o dia, o que afeta a precisão de todos os modelos. Logo, a diferença do volume de dados por período de tempo é algo importante a se considerar quando for realizar a previsão da mobilidade humana.

Como trabalho futuro pretende-se investigar novas estratégias de combinação de dados heterogêneos que permitam a identificação (ou pelo menos a inferência da presença) de um mesmo usuário em múltiplas coleções. Este é um grande desafio técnico uma vez que essas coleções tipicamente são obtidas de forma independente. Pretende-se também adaptar o MobDatU para prever o volume de pessoas que estarão em uma determinada região em um certo período. Tais previsões são úteis para suportar decisões de gerenciamento e planejamento urbano.

Referências

Allamanis, M., Scellato, S., and Mascolo, C. (2012). Evolution of a location-based online social network: Analysis and models. In *Proceedings ACM Conference on Internet Measurement*.

Balcan, D., Colizza, V., Gonçalves, B., Hu, H., Ramasco, J. J., and Vespignani, A. (2009). Multiscale mobility networks and the spatial spreading of infectious diseases. *Proceedings of the National Academy of Sciences*, 106(51):21484–21489.

Bui, N., Bui, N., Michelinakis, F., Michelinakis, F., and Widmer, J. (2014). A model for throughput prediction for mobile users. In *Proceedings 20th European Wireless Conference;*.

Candia, J., González, M. C., Wang, P., Schoenharl, T., Madey, G., and Barabási, A.-L. (2008). Uncovering individual and collective human dynamics from mobile phone records. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224015.

Cho, E., Myers, S. A., and Leskovec, J. (2011). Friendship and mobility: User movement in location-based social networks. In *Proc. 17th ACM International Conference on Knowledge Discovery and Data Mining*.

Dong, W., Duffield, N., Ge, Z., Lee, S., and Pang, J. (2013). Modeling cellular user mobility using a leap graph. In *Proc. 14th International Conference on Passive and Active Measurement*.

González, M. C., Hidalgo, C. A., and Barabási, A.-L. (2008). Understanding individual human mobility patterns. *Nature Publishing Group*, 453.

Jeung, H., Liu, Q., Shen, H. T., and Zhou, X. (2008). A hybrid prediction model for moving objects. In *Proc. IEEE 24th International Conference on Data Engineering*.

Lee, K., Hong, S., Kim, S. J., Rhee, I., and Chong, S. (2009). Slaw: A new mobility model for human walks. In *Proc. INFOCOM*.

Lee, K., Hong, S., Kim, S. J., Rhee, I., and Chong, S. (2012). Slaw: Self-similar least-action human walk. *IEEE/ACM Transactions on Networking*, 20(2):515–529.

Munjal, A., Camp, T., and Navidi, W. C. (2011). Smooth: A simple way to model human mobility. In *Proc. 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*.

Musolesi, M. and Mascolo, C. (2007). Designing mobility models based on social network theory. *SIGMOBILE Mobile Computing Communications Review*, 11(3).

Nguyen, T. and Szymanski, B. K. (2012). Using location-based social networks to validate human mobility and relationships models. In *Proc. International Conference on Advances in Social Networks Analysis and Mining*.

Noulas, A., Scellato, S., Lambiotte, R., Pontil, M., and Mascolo, C. (2012a). A tale of many cities: universal patterns in human urban mobility. *PloS one*, 7(5):e37027.

Noulas, A., Scellato, S., Lathia, N., and Mascolo, C. (2012b). A random walk around the city: New venue recommendation in location-based social networks. In *Proc. International Conference on Social Computing*.

Rhee, I., Shin, M., Hong, S., Lee, K., and Chong, S. (2008). On the levy-walk nature of human mobility. In *Proc. INFOCOM*.

Song, C., Qu, Z., Blumm, N., and Barabási, A.-L. (2010). Limits of predictability in human mobility. *Science*, 327(5968):1018–1021.

Twitter (2013). Twitter streaming api (dev.twitter.com/streaming/overview).

Xavier, F. H. Z., Silveira, L. M., Almeida, J. M. d., Ziviani, A., Malab, C. H. S., and Marques-Neto, H. T. (2012). Analyzing the workload dynamics of a mobile phone network in large scale events. In *Proc. First Workshop on Urban Networking*.

Zheng, Y. and Xie, X. (2011). Learning travel recommendations from user-generated gps traces. *ACM Transactions on Intelligent Systems Technology*, 2(1).

A QoE-aware Mechanism to Improve the Dissemination of Live Videos over VANETs

Carlos Quadros[123], Aldri Santos[2], Mario Gerla[3], Eduardo Cerqueira[13]

[1]Faculty of Computer Engineering and Telecommunication – UFPA
[2]Wireless and Advanced Networks (NR2) - Dept. of Informatics – UFPR
[3]Computer Science Dept. – University of California, Los Angeles (UCLA)

{quadros,cerqueira}@ufpa.br, aldri@ufpr.br, gerla@cs.ucla.edu

Abstract. *Real-time video dissemination over Vehicular Ad-hoc NETworks (VANETs) is challenging due to its strict quality requirements, dynamic network topology, and broadcast environment. Although Statistical Routing Protocols (SRPs) have been applied by using positioning and Quality of Service (QoS) parameters, they are not enough to obtain satisfactory Quality of Experience (QoE) levels in multi-hop video dissemination due to different requirements of multimedia services. This paper introduces the QOe-aware REceiver-based (QORE) mechanism to improve the dissemination of real-time videos in VANETs. QORE employs a QoE-driven Unequal Error Protection (UEP) scheme, namely Interleaving, which mitigates the effects of frame loss by spreading out the bursty losses. Further, QORE can be integrated with SRPs to offer QoE-driven parameters for the relay node selection and backbone maintenance. Thus, nodes decide for themselves to retransmit further the video sequences, enhancing the capacity of the system in delivering QoE-aware live videos. QORE was added to a straightforward Distance-based SRP, named DQORE. Results show the gains of DQORE in relation to SRPs, achieving video dissemination with QoE support, less routing overhead, and robustness in V2V VANETs.*

1. Introduction

Nowadays, people are increasingly spending more and more time in their vehicles. This has encouraged industries and manufacturers to provide drivers and passengers with a wide scope of novel real-time multimedia services, ranging from safety and security traffic warnings to live entertainment and advertising videos [Kakkasageri and Manvi 2014]. In this way, efficient on-road live video dissemination support in Vehicular Ad-hoc NETworks (VANETs) has become a trend and, at the same time, necessary for the future of the Intelligent Transportation System (ITS) [Gerla et al. 2014].

To meet this demand, vehicles have employed flooding approaches extensively as data delivery mechanism. Although, the simple flooding of data within the network leads to an explosive growth of traffic resulting in collisions and congestions, i.e., the broadcast storm problem [Torres et al. 2014]. In addition, due to the high vehicle mobility, the delivery of real-time video content, while maximizing the user's Quality of Experience (QoE), becomes an intricate task and has not been taken into account [Felice et al. 2014]. While Quality of Service (QoS) focuses only on packet-based management and delivery statistics, QoE levels of the received videos must be measured by the subjective acceptability of the users, the latter being key to the safety broadcast of traffic warning videos.

The design of a reliable and robust dissemination approach for real-time video dissemination over VANETs with QoE support is not a straightforward task [Quadros et al. 2014]. Due to ad-hoc nature and the highly dynamic topology, when very close neighbor vehicles decide to transmit the same packets, they can interfere with each other and lead to an unnecessary growth of traffic of data, causing different impacts on the video quality and wasting network resources. Thus, the multi-hop routing service must be aware of QoE requirements and network conditions to recover or maintain video quality with a low overhead and with high reachability for the nodes in the flooding area.

On the other hand, most of the current routing protocols use information exchanged by neighbor vehicles to make routing decisions. However, end-to-end routes suffer from frequent interruptions, and the currently available local topology is not always accurate [Slavik et al. 2014]. Statistical Routing Protocols (SRPs) allow improvements on network performance by making a distributed hop-by-hop routing decision [Torres et al. 2014]. In SRPs the selection of relay nodes is carried out by receivers (receiver-based), without requiring static end-to-end routes, and allowing transmissions even in case of topology changes. However, the existing SRPs do not efficiently explore vehicle positioning, network, and QoE-driven parameters. Thus, a QoE-aware mechanism, that employs information such as hierarchy of frames [Yao et al. 2014] and estimation of video distortion (caused by probability of occurrence of loss and loss burstiness) [Tao et al. 2008] can be coupled as a QoE-pointer to support real-time monitoring and forwarding decision at the routing level [Aguiar et al. 2014].

QoE-driven Unequal Error Protection (UEP) schemes can further enhance the advantages of SRPs[Claypool and Zhu 2003, Li et al. 2014], since they use information from the application-layer (video characteristics and requirements) to minimize the effects of packet loss and achieve better video distribution. These schemes allow multimedia dissemination with QoE support even in the presence of dynamic topologies [Rosário et al. 2014, Immich et al. 2014]. This way, the Interleaving UEP scheme improves the QoE levels of live video flows by merging frames of the original video sequence without increasing packet redundancy, thus avoiding the broadcast storm problem. Therefore, a straightforward SRP coupled with a QoE-aware mechanism and an appropriate QoE-driven UEP scheme (Interleaving) in a unified approach, can contribute to video dissemination with better QoE assurance in dynamic topologies.

This paper proposes a QoE-aware mechanism to support the delivery of real-time videos over V2V VANETs, named QOe-aware REceiver-based (QORE) mechanism. It combines application parameters (e.g., different frame importance, frame position, and video distortion estimation), positioning information, and the QoE-driven Interleaving scheme to establish trade-offs between quality and required hops. QORE performs a self-organized use of relay nodes and can be easily integrated with SRPs, maintaining the packet delivery ratio, reacting well to dynamic environments and enhancing or at least maintaining the QoE level of the disseminated videos when compared to non-QoE-driven schemes. QORE was added to a straightforward Distance-based SRP and was evaluated.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces the QORE mechanism and the coupling of QORE with the Distance-based SRP (called DQORE). Simulation setup and results comparing pure SRPs and DQORE are presented in Section 4. Lastly, conclusions are summarized in Section 5.

2. Related Work

There are three fundamental methods that fit into the category of SRPs (i.e., the next relay nodes are chosen through a distributed backoff phase, by comparing a locally measured value with a threshold value): Counter-Based (CB), Distance-Based (DB), and Location-based (LB) methods [Torres et al. 2014]. The DB method uses the distance to the farthest 1-hop neighbor from whom the packets has been sent as a proxy for rebroadcasting. The LB method, in turn, employs sharing of positional information to allow retransmissions in the uncovered area. Lastly, in the CB method, nodes simply count the number of times that each packet is received during the backoff to count the number of neighbors that so far have retransmitted the packets. These methods have been used to design many protocols including some using hybridization with topological approaches.

In [Mohammed et al. 2009], authors published improved versions of the above statistical methods by adding parameters related to density to further reduce the broadcast storm problem. In [Slavik and Mahgoub 2013] authors proposed the Distribution-Adaptive Distance with Channel Quality (DADCQ) protocol. DADCQ is a DB SRP that is adaptive to distribution pattern and channel quality for multihop V2V broadcast. However, all of these approaches do not take into account video-awareness to the video dissemination, i.e., they do not ensure QoE in the received video streaming. Furthermore, it is important to evaluate new video-based routing schemes based on QoE metrics.

In [Rosário et al. 2014], authors proposed the Link quality and Geographical Opportunistic Routing (LINGO) protocol, a receiver-based protocol that uses a multi-criteria scheme to compute the backoff timer and employs a UEP scheme, namely Forward Error Correction (FEC). Similarly, in [Di Felice et al. 2013], authors presented a geocast and contention-based protocol for multimedia dissemination in VANETs, called Dynamic Backbone Assisted (DBA) MAC protocol. In DBA-MAC, the formation of the backbones also uses a multi-criteria approach, such as link quality, vehicles location, and speed. Despite LINGO and DBA-MAC consider the link quality for routing decisions, these protocols do not take into account QoE-based criteria for the relay node selection and backbone maintenance. Further, these approaches use only one sample of packet to define the best forwarder nodes, which can cause false-positive results on the link quality measurement. Although the usage of a UEP scheme in LINGO, the use of FEC may worsen the broadcast storm problem due to increase of redundant packets in the network.

In [Slavik et al. 2014], authors introduce the Distance-to-Mean (DTM) method, which extends the DB method, where nodes favor rebroadcasting when they cover a large amount of physical area that has not been covered by neighboring nodes. Similarly, in [Torres et al. 2014], authors propose Automatic Copies Distance-Based (ACDB), an improved beaconing flooding scheme that extends the CB method to cope with variable vehicle density situations. Besides ACDB uses the Peak signal-to-noise ratio (PSNR) QoE-metric to assess the received videos, a basic limitation of these protocols consists of their reliance on a single criterion to compute the backoff phase, reducing the network reliability for long data transmission, such as live video streaming. Moreover, not only PSNR but also, other real QoE experiments and more sophisticated objective metrics, as shown in Section 4, must be applied to assessment of received videos, since PSNR by itself, does not correlate well with the subjective acceptability of the users [Aguiar et al. 2014]. DTM and ACDB are used as comparison protocols in this work.

From our analysis, a SRP is a promising solution for broadcast in VANETs, since vehicles do not need to flood messages proactively, avoiding broadcast storms. In addition, the existing SRPs do not efficiently combine location information, QoE-awareness, and a UEP scheme for video dissemination. All of these key features are not offered in a unified SRP so far, thus existing proposals lack of robustness and QoE-awareness.

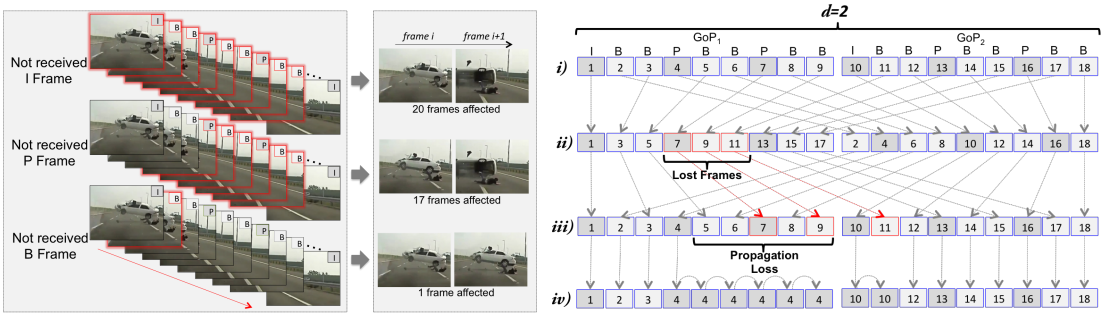
3. The QOe-aware REceiver-based (QORE) mechanism

This section presents the QORE mechanism to improve dissemination of live videos in multi-hop V2V VANETs. It considers a receiver-based approach, where no neighboring information exchange is required. QORE works jointly with an underlying SRP and aims to select relay nodes with high reachability, i.e., nodes that will cover as many destinations in a physical area as possible avoiding unnecessary routing overhead. Further, QORE allows video dissemination with high quality and low impact from the user's point-of-view even in the presence of dynamic topology scenario changes. Depending on the SRP, other different parameters can be incorporated aiming at a more robust decision process. QORE works in three phases, namely QoE-driven Unequal Error Protection (QUEP), Distributed Backoff-based Forwarding (DBF), and Persistent Multi-hop Forwarding (PMF). In the QUEP phase, we employ the video Interleaving method [Claypool and Zhu 2003], where the source node first re-sequences video frames before delivering (to balance the loss of packets) and returns to their original order at the receiver nodes. Upon re-sequencing the video frames in QUEP, in DBF, the source node starts the video flooding and the relay node candidates compete to choose which nodes will participate in the PMF phase. Thus, the chosen nodes retransmit the video sequences further, according to positioning and QoE-driven parameters. The PMF phase provides contention-free forwarding of the video streaming, exploiting the previously built multi-hop paths, and allowing dynamic changes to other paths (relay nodes) in case of link failures and loss of quality. We will detail each phase in the following subsections.

In our network model, let's suppose a scenario of live video dissemination, so that in case of accidents or disasters, vehicles or first responder teams coming toward the crashed area receive videos of accident. Thus, we consider k vehicles containing, each one, an identifier ($i \in [1, k]$), moving over an urban multi-lane grid-area. The combination of those nodes configures a graph of vehicular topology $G(V, E)$, where vertices $V = \{v_1, v_2, ..., v_k\}$ mean a finite subset of k nodes, and edges $E = \{e_1, e_2, ..., e_m\}$ mean a finite set of asymmetric wireless links between them. We denote a subset $N(v_i) \subset V$ as all 1-hop neighbors within the radio range of a given node v_i. Further, each node v_i has an IEEE 802.11p-compliant radio transceiver, through which it can communicate with $N(v_i)$, a GPS (to location awareness and synchronism in time), a multimedia encoder/decoder, and a transmission buffer (*TB*) with a maximum queue capacity (TB_{Max}).

3.1. QoE-driven Unequal Error Protection (QUEP)

Before transmitting, video sequences are compressed by the Motion Picture Expert Group (MPEG) standard in Group of Pictures (GoPs) composed of three frame types, namely I-, P-, and B-frame. Frames between two I-frames define a GoP. I-frames are self-contained, however, to encode and decode P- and B-frames, the previous I-frame and/or P-frames in the same GoP are needed. The compression rate of P-frames is higher than the rate of I-frames, and the compressed rate of B-frames are the highest of all three. Thus, if



(a) (b)

Figure 1. (a) Different priority of frames. (b) Interleaving scheme.

an I- or P-frame is lost, all frames thereafter in the GoP become un-decodable, i.e., the same degree of packet loss may cause severe quality degradation or may pass unnoticed, depending on which frame types are affected. Fig. 1a shows the different degrees of importance for the user's perception in each frame type for a GoP with size 20. I-frames are the most important ones from the human point-of-view. For a single I- or P-frame lost, there is error propagation through frames until the end of the GoP. Though, for a single B frame lost, the impact is not noticed visually, since no others frames are affected.

For the QUEP phase, we consider the Interleaving scheme as QoE-driven UEP mechanism. Because of its modular design, QORE can be easily adapted to use other UEP schemes, since these techniques are carried out before the transmission phase (DBF). We opted for Interleaving due to its low overhead, essential in broadcast environments. This scheme assumes that a better quality from the user's perspective can be achieved by spreading out bursty packet losses in a media flow, i.e., small gaps degrade quality less than a big gap in a multimedia flow. Thus, by using a distance (d), given by the interleaving algorithm, whenever a source node (VS) has a video flow ($VF = \{g_1, g_2, ..., g_n\}$) with n GoPs (g) to send, it first interleaves $d \times |g|$ consecutive frames, i.e., d represents the number of merged GoPs. After that, VS starts the DBF phase, by broadcasting the interleaved frames as flooding. Upon reaching receivers, these frames are then reconstructed back to their original order. If consecutive loss occurs in the interleaved stream during transmission, big gaps can be spread out into several small gaps [Li et al. 2014].

Interleaving is based on the Frame-Copy error concealment technique, where the last well-received frame replaces lost frames. Fig. 1b shows the interleaving mechanism for two ($d = 2$) consecutive GoPs ($|g| = 9$) (***i)***). Upon losing 3 consecutive frames (4th, 5th, and 6th) in the 1st interleaved GoP during the broadcast (***ii)***), the Interleaving scheme breaks the gap into small gaps (***iii)***). Through Frame-Copy, the reconstructed GoPs at receivers are recovered by repeating the previous well-received frame (***iv)***). Otherwise, without interleaving, in case of lost of the 4th, 5th, and 6th frames in ***i)***, the gap of lost frames would be only one, but with a much larger size, since the first P-frame (4th frame) is necessary for the reconstruction of the remaining frames in the GoP. Thus, this division of bursty losses in small gaps becomes less noticeable from the user's point-of-view.

3.2. Distributed Backoff-based Forwarding (DBF)

In the DBF phase, the interleaved video flows are disseminated through a contention distributed stage, where VS starts the video transmission and relay nodes (VR) compete among themselves to choose which nodes will participate in the PMF phase. By using

QoE-driven parameters, called QoE-indicators, QORE defines the best VRs in each hop. Once VS begins to capture a given VF, it starts the dissemination in a multi-hop fashion, i.e., VS broadcasts video packets p in a *Time Window*, denoted by $W(VF_i) \subset VF_i$, to all the neighbors of VS ($N(VS)$). The Algorithm 1 presents the generic process of DBF.

Upon receiving a $W(VF_i)$ from a sender node v_a, the nodes $N(v_a)$ have their own and v_a variation of position information, extracted from the packet headers. Thus, a given node $v_b \in N(v_a)$ can easily determine when it is located within the set of nodes in the Forwarding Zone (FZ) of v_a ($FZ(v_a) \subset N(v_a)$) (Line 1 of Algorithm 1), that corresponds to an angle α between the line of movement orientation of v_a and v_b. FZ becomes important, since it limits the selection of VRs to a given sector, avoiding loops and vehicles going to the opposite direction of the crash area. We have defined $\alpha \leq 90^\circ$ to FZ [Wang et al. 2014]. VR candidates have information of the total number of hops traversed N_{Hops} of $W(VF_i)$, i.e., for a $W(VF_i)$ exceeding a predetermined threshold N_{HopsT}, no further rebroadcast occur (Line 1 of Algorithm 1). We have defined $N_{HopsT} = 10$, since for a range of 250m in each hop, we have a medium-long distance of 2.5km approximately, as required in many rescue/disaster VANET scenarios [Felice et al. 2014].

Algorithm 1 DBF phase

When a given node $v_b \in N(v_a)$ receives broadcasted packets ($W(VF_i) = \sum_{k=1}^{n} p_k$) from a node v_a:
1: **if** $v_b \in FZ(v_a)$ **and** $N_{Hops}(W(VF_i)) < N_{HopsT}$ **then**
2: **if** $\exists!\ p_k \in W(VF_i)$ **and** $p_k \in TB_{v_b}$ **then**
3: Drop $W(VF_i)$ from TB_{v_b}
4: **return**
5: **else**
6: Compute $FF(v_b)$ (Eq. (6)) **and** Start $BackoffTimer(v_b)$ (Eq. (1))
7: **while** $BackoffTimer(v_b) \neq 0$ **do**
8: **if** Overhear ! $p_k \in W(VF_i)$ **then**
9: Cancel $BackoffTimer(v_b)$
10: **if** $FF > FF_t$ **and** $\angle VR_i v_a v_b \geq \phi$ **then**
11: Rebroadcasts $W(VF_i)$ **and** $v_b \leftarrow VR_{i+1}$
12: **else**
13: Drop $W(VF_i)$ from TB_{v_b} **and** Cancel any new rebroadcast of $p_k \in W(VF_i)$
14: **return**
15: **end if**
16: **end if**
17: **end while**
18: Rebroadcasts $W(VF_i)$ **and** $v_b \leftarrow VR_i$
19: **end if**
20: **else**
21: Drop $W(VF_i)$
22: **return**
23: **end if**

If $W(VF_i)$ contains only new received packets (Line 2 of Algorithm 1), nodes located within $FZ(v_a)$ apply the *Fit Function* (FF) (Eq. (6)) prior to relay the packets, conversely nodes outside this area drop the received packets from *TB*. The use of FF allows VR candidates to mitigate the number of retransmissions inside FZ by choosing only the best VRs $\in FZ(v_a)$. The value of FF $[0, FF_{max}]$ depends on parameters of the underlying SRP, such as positioning (Subsection 3.2.2), and QoE-driven parameters, as shown in Subsection 3.2.1. Thus, after calculating the FF, VR candidates replace the

location of v_a by their own locations in the packet header, set a $BackoffTimer$ according to Eq. (1), and wait for the timeout to rebroadcast the buffered packets ($W(VF_i)$). It is easy to see that nodes with higher values of FF are mapped to smaller $BackoffTimer$ sizes, and thus having higher probability to win in the DBF phase.

$$BackoffTimer = CW_{Max} - FF \cdot (CW_{Max} - CW_{Min}) \quad (1)$$

Where CW $[CW_{Min}, CW_{Max}]$ is the size of the *Contention Window* in the 802.11p standard. In a completely distributed manner, the node that generates the smallest $BackoffTimer$ rebroadcasts $W(VF_i)$ first and it is selected as VR (line 18 of Algorithm 1). Thus, QORE provides video dissemination via multiple VRs. Moreover, as expected in the IEEE 802.11p standard, VR candidates are able to sense the channel during the $BackoffTimer$, so that they will cancel their countdown timers in case of overhearing transmissions of any $p \in W(VF_i)$ from other nodes. In this case, other nodes in the $FZ(v_a)$ compare their calculated FF to a threshold value (FF_t) and the angle between VR_i, v_a, and v_b to a threshold angle $\phi = 45^\circ$ ($\phi = \alpha/2$), so that these VR candidates can retransmit further the received packets. The FF_t and ϕ values are directly related to the reachability of the underlying SRP. If the calculated FF is bigger than FF_t, and the angle between the previous selected VR and v_b is bigger than ϕ, the VR candidate proceeds to retransmit $W(VF_i)$ itself, otherwise it remains silent.

The QORE mechanism aims to select relay nodes that have better video quality on the user's perspective. Thus, we used a fixed $FF_t = 0.4$ [Slavik et al. 2014]. As outlined in Subsection 3.2.2, to avoid duplicated packets in the same geographic area, the DB method establishes that the farthest nodes (from the previous sender node) rebroadcast further the same $W(VF_i)$. Moreover, through a *passive acknowledgment* approach [Rosário et al. 2014], v_a is also able to overhear the further relaying of $W(VF_i)$ and, thus, concluding that it was successfully received by other nodes in the $FZ(V_a)$, allowing QORE to reduce the acknowledgments in the MAC-layer.

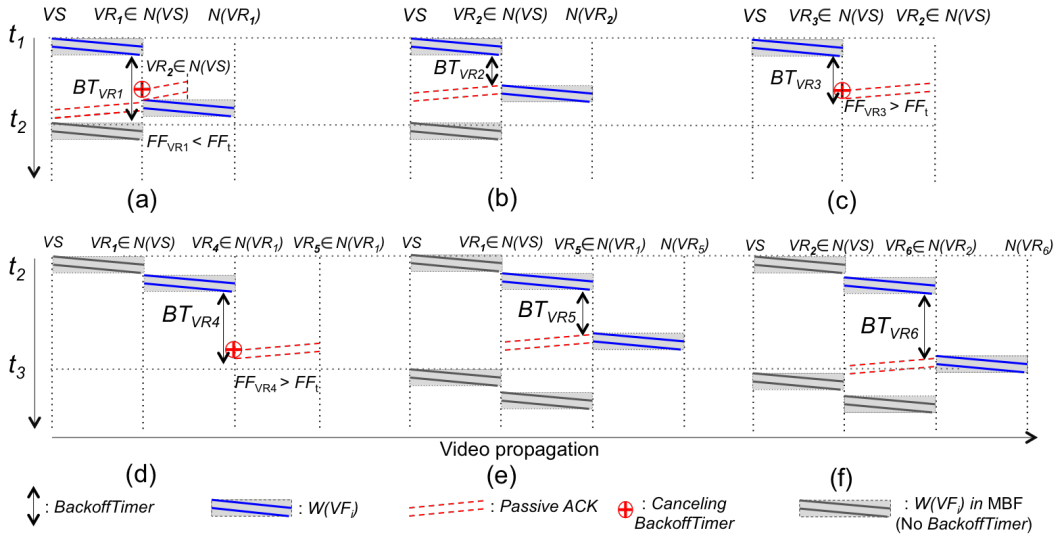


Figure 2. DBF phase for the 1st hop (a-c), and for the 2nd hop (d-f)

Fig. 2 shows an overview of the DBF phase. Suppose that $N(VS) \cap FZ(VS) =$

$\{VR_1, VR_2, VR_3\}$, i.e., nodes VR_1, VR_2, and VR_3 are located in the Forwarding Zone of the source vehicle. In this example, VR_2 forwarded a given $W(VF_i)$ first (t_1 in Fig. 2b) and the neighboring nodes (VR_1 and VR_3) overhear this transmission (t_1 in Figs. 2a and c). As a result, these nodes cancel their $BackoffTimers$ and check out whether their calculated FF are smaller than the FF_t. The neighboring node which has a FF smaller than FF_t, proceed to retransmit $W(VF_i)$ (t_2 in Fig. 2a), otherwise it drops $W(VF_i)$ from its TB and remains silent (line 13 of Algorithm 1 and t_2 in Fig. 2c).

In t_2, for the second hop (Figs. 2d-f), suppose that $N(VR_1) \cap FZ(VR_1) = \{VR_4, VR_5\}$ and $N(VR_2) \cap FZ(VR_2) = \{VR_6\}$, i.e., VR_4 and VR_5 are located in the FZ of VR_1; and only VR_6 is located in the FZ of VR_2. Initially, VR_5 wins the contention phase with the smallest $BackoffTimer$ and forwards $W(VF_i)$ first. Then, VR_1 uses the transmitted packets as *passive acknowledgement*. Meanwhile, the second VR candidate (VR_4, from Fig. 2d) overhears VR_5 transmitting $W(VF_i)$, cancels its $BackoffTimer$ and checks out whether its calculated FF is smaller than the FF_t. As FF_{VR4} is bigger than FF_t, VR_4 remains silent. Finally, in t_3 of Figs. 2d and e, VS transmits subsequent packets to VR_5 and VR_6 by VR_1 and VR_2 without requiring $BackoffTimers$, i.e., it switches to the PMF phase, detailed in Subsection 3.3.

3.2.1. QoE-Indicators

An MPEG video sequence is composed of frames with different degrees of importance for the user's perception, as seen previously. In a 24 fps MPEG-4 video, if packets from one I-frame are lost, the video will be degraded in at least 0.75 s. Also, the loss of P-frames at the beginning of a GoP causes a higher video distortion than loss at the end of a GoP. However, the loss of B-frames does not impact heavily on the user's perception. By considering the importance of each video frame, as well as the P-frame position within the GoP, QORE prioritizes frames with a greater impact on the average video distortion (σ_s^2), as opposed to those with lower QoE impact. Thus, it assigns different weights to slices (s) of packets belonging to each frame as modeled by Eq. (2):

$$\sigma_s^2 \propto \begin{cases} \frac{\alpha_1(R_M - R_I)}{R_M} & \text{if s} \in \text{I-frame} \\ \frac{\alpha_2}{(2^{T-1}-1)R_M} \sum_{i=1}^{T-1} 2^{T-1-i}(R_M - R_{P_i}) & \text{if s} \in \text{P-frame} \\ \frac{\alpha_3(R_M - R_B)}{R_M} & \text{if s} \in \text{B-frame} \end{cases} \quad (2)$$

Where $T-1$ is the number of P-frames per GoP, R_I, R_{P_i}, and R_B mean the I, P (with position i in the GoP), and B-frame received rate in $W(VF_i)$, respectively. R_M is the maximum data-rate supported by the radio transceiver of each vehicle, e.g., for a DCMA-86P2 802.11p Wi-Fi card, $R_M = 6$ Mbps if $P_{rx} >$ -93dbm [Di Felice et al. 2013]. The parameters α_1, α_2, and α_3 are weighting factors, where $\sum_{i=1}^{3} \alpha_i = 1$.

The distortion model proposed by Shu Tao [Tao et al. 2008] considers the impact caused by the loss of single slices of a frame from a video stream. Thus, for a given video frame structure and a probability of occurrence of loss, Eq. (3) defines an overall distortion value for the whole stream, where s and L are the number of slices per video packet and the number of packets per frame, respectively obtained from the standard video codec configurations and packetization. The parameter $\overline{n}$ represents the loss burstiness

($1.06 \geq \overline{n} \geq 1$ for Bernoulli losses, depending on the aggressiveness of the burst errors). The attenuation factor γ ($\gamma < 1$) accounts for the effect of spatial filtering, and varies as a function of the video characteristics and decoder processing. P_e is the probability of loss events (of any length) in the video stream. Both, γ and P_e are given by the effect of the loss pattern experienced by the video stream and the codec's error concealment technique. Finally, the Mean Square Error (MSE) distortion $\overline{D}$ provides a QoE-estimate by using a non-linear relation that measures the video quality level by comparing distortions caused by slice losses [Tao et al. 2008], according to each frame type, denoted in Eq. (4):

$$\overline{D} = sL \cdot \overline{n} P_e \cdot \sigma_s^2 \cdot \left(\frac{\gamma^{-t+1} - (T+1)\gamma + T}{T(1-\gamma)^2} \right) \tag{3}$$

$$C_{QoE}^{v_b}(W(VF_i)) = \frac{1}{1 + exp(b_1 \cdot 10 \cdot log_{10}(255^2/\overline{D}) - b_2)} \tag{4}$$

Where, b_1 is the slope of the QoE mapping curve and b_2 is the central point. By considering 40 dB as the highest video quality, and the lowest video quality for values below 20 dB, the values of b_1 and b_2 are given by 0.5 and 30, respectively. Based on the average distortion caused by losses in the different frame types in $W(VF_i)$ it is possible to VR candidates to compute a higher FF to nodes receiving the most important packets.

3.2.2. Integration of the DBF phase of QORE with the DB SRP

Here we present the QORE mechanism operating together with an underlying routing protocol. To assess QORE functionalities, we develop and adapt its DBF phase jointly with a straightforward broadcast protocol built using the DB method, called DQORE protocol. The DB method calculates coverage through the distance (C_D) from the VR candidate to the previous sender node. When C_D is small, it means the VR candidate is close to the last sender, indicating it should not favor rebroadcasting. Local positional information is used in the DB method to calculate C_D in VR candidates. Thus, the spatial distance is defined in Eq. (5) for a VR candidate positioned at (x, y) and a previous sender node located at $(\overline{x}, \overline{y})$ and normalized to a value between zero and one by dividing by the maximum transmitting range (R). Hence, VR candidates with large geographical distance from previous sender node generate higher FF values.

$$C_M = \frac{1}{R}\sqrt{(x-\overline{x})^2 + (y-\overline{y})^2} \tag{5}$$

To add QORE with the DB SRP, we establish two criteria as input to FF (Eq. (6)): vehicle positioning and QoE-indicators, allowing a cross-layer VR selection in addition to the only positioning parameters of the pure DB method. As defined in Eq. (5), the DB method does not exchange messages containing location or mobility information. Thus, DQORE also maintains the statistical and receiver-based features of the DB method when determining the best VR options. Further, QORE can be coupled with other SRPs by simply changing the parameters in the DBF phase for the VR selection process. It might be suitable for link quality-based, stochastic-based, or CB SRPs. Depending on the routing strategy, the QORE steps in the DBF phase can be easily adapted.

3.3. Persistent Multi-hop Forwarding (PMF)

As video transmissions are often long (e.g., 20 s), whenever a given node wins the DBF phase, VS transmits video packets explicitly without any additional delay and in a backbone fashion (t_2 in Figs. 2e and f). Thereby, QORE reduces additional delays and packet duplication from the DBF phase by introducing the Persistent Multi-hop Forwarding (PMF) phase. During the transmission, the video content should be delivered even in the presence of node failures or channel variations. QORE detects routing failures, providing a smoother backbone management. In particular, QORE considers that every node that composes the video dissemination backbone $P(VS, VD)$ should perceive whether it is still a reliable or valid route to transmit packets. This is achieved by receiving reply messages. We define a control packet, called Peer Quality Message (PQM), which contains the $\overline{D}$ perceived by each forwarder. Thus, if a given VR_2 receives a video flow from VR_1, it must compute $\overline{D}$ perceived in each $W(VF)$ and send a $PQM_{W(VF)}$ to VR_1.

Any built backbone returns to the DBF phase, when it detects that the video quality falls below a predefined video distortion threshold ($\overline{D}_t$). Further, any node that composes a transmitting backbone considers that the route is not valid anymore, as long as it does not receive any reply message from its previous VR within a certain period of time, i.e., timeout $= 0.5s$. Hence, it returns to the DBF phase to re-establish a new backbone.

Upon computing C_{QoE} and C_D, each VR contains the calculated criteria, i.e., $D = \{C_{QoE}, C_D\}$ ($|D| = 2$). Thus, considering the different weights $\omega_p | \sum_{p=1}^{|D|} \omega_p = 1$, Eq. (6) calculates FF by multiplying the values d_p in D and the weights of evaluation criteria, similarly to others multi-criteria approaches [Rosário et al. 2014]. From Eq. (6), other criteria can be added to FF of QORE, depending only on the underline SRP.

$$FF = \sum_{p=1}^{|D|} (d_p \times \omega_p) \tag{6}$$

4. Performance Evaluation

This section shows the methodology and metrics used to evaluate the transmitted video flows, where DQORE is compared to main related works. To establish a relevant scenario, we have considered a 5 square kilometers of West Los Angeles from OpenStreetMap, which was imported into SUMO (Simulation of Urban MObility). It allows us to reproduce the desired vehicle movements and V2V interactions according to empirical data. Further, in our simulations, vehicles move with a speed ranging from 6 to 20 m/s, where each one holds an IEEE 802.11p (5.89 GHz, 6 Mbps) radio with about 250 m transmission range and a TB_{Max} of 30 pkts. By following the approach proposed in [Torres et al. 2014], we scheduled an accident situation, so that when VS perceives the accident, it starts the dissemination of VF. Each VF must be received by vehicles at a distance lower than 2.5 km from VS, providing limitation of hops, as proposed in [Di Felice et al. 2013]. The Nakagami Fading Channel was used as propagation model.

Aiming for realistic results, we have used EvalVid - A Video Quality Evaluation Tool-set that allows us evaluating the video quality. Thus, we have conducted the experiments by transmitting real MPEG-4 sequences (720 x 480 pixels) lasting approximately 25 s, available in [Video Sequences 2014], with 768 kbps and 24 fps, internal GoP struc-

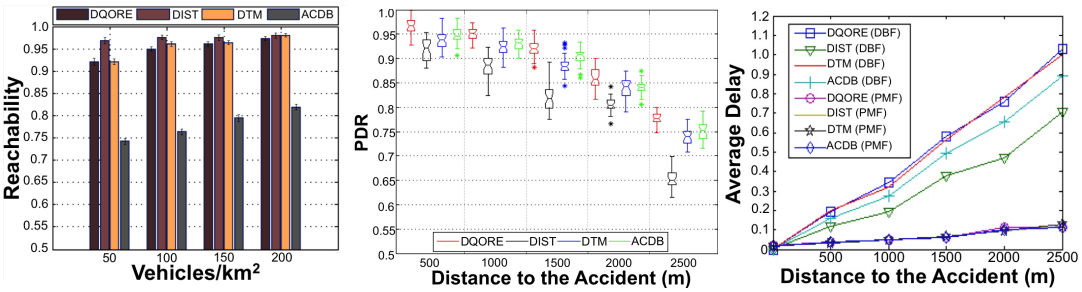
ture (size 20) configured as two B-frames for each P-frame, and interleaving distance $d = 2$. The videos and the road/vehicle features were added into Network Simulator 2.33.

To demonstrate the impact of DQORE in broadcasting QoE-aware video flows in VANETs, we used a straightforward SRP built from the DB method (named DIST), DTM [Slavik et al. 2014], and ACDB [Torres et al. 2014] for comparison. The DTM and ACDB protocols use DB and CB methods, respectively. In DTM, the farther away the node to the spatial mean, the shorter the $BackoffTimer$. ACDB uses the density information provided by a beaconing approach to dynamically adjust a counter and the maximum $BackoffTimer$ before rebroadcasting packets. These protocols were adjusted with the PMF phase to reduce the contention phase: once a vehicle successfully transmits video packets, its timer for the next packets will be zero. We introduced these improvements because the standard protocols, as they were, did not represent a fair comparison.

The I-, P,- and B-frame weights (α_1, α_2, and α_3) affect the QORE performance. We have conducted independent empirical evaluations and we concluded that $\alpha_1 = 0.65$, $\alpha_2 = 0.3$, and $\alpha_3 = 0.05$ give the best C_{QoE} results. Further, the weights for each criterion ω_1 and ω_2 were fixed in 0.6 and 0.4, respectively, which allow QORE achieve the best trade-off between lowest number of hops and enough QoE-indicators to an acceptable video quality. In addition, we set CW_{Max} to 100 ms, $W(VF)$ to 80 ms, and $\overline{D}_t$ to 0.75.

The above protocols were assessed by Packet Delivery Rate (PDR), average delay, and reachability, which shows the average fraction of nodes that receive the broadcasted videos. Since measuring the human experience is key for our work, QoE-based measurements were carried out with Structural SIMilarity (SSIM) and Mean Opinion Score (MOS). SSIM is a objective QoE metric that measures the structural distortion of the video to obtain a better correlation with the user's perspective. For the subjective experiments (MOS), an Android application [Felice et al. 2014] was used (following the ITU-R rules) to playback the transmitted videos and collect their evaluations. We used the Single Stimulus method of ITU-R BT.500, with 30 subjects (ranging from 18 to 40 years old), where after watching a video, each viewer assesses the video quality level by selecting a score ranging from 1 (poor) to 10 (excellent). The distorted videos were played on a tablet (8.4-inch) placed in the back of the headrest of a car. The results are shown with variations in the number of vehicles (50 - 200 veh/km^2) and distance to the crash area (500 - 2500 m), being an average of 35 simulations (95% confidence level). Each simulation lasts 500 s, where, a VS sends a VF at any time after the initial 100 s and before the last 100 s. At receivers, the decoder uses Frame-Copy as the error concealment technique.

Regarding network performance in reachability, PDR, and average delay, Figs. 3a-c show the performance results for the four simulated protocols. From Fig. 3a, DQORE, DIST, and DTM notably outperform ACDB in terms of reachability. This is because ACDB does not uses positioning parameters to relay node selection (CB SRP). On average, DQORE and DTM increase the reachability by 18.6% and 18.9% compared to ACDB, respectively, while DIST increases reachability by 2.4% and 2.1% over DQORE and DTM, respectively. The highest reachability of DIST occurs due to the selection criteria of DQORE and DTM, where the forwarding vehicles are (potentially) close vehicles that experience more stable link connectivity, allowing a higher PDR, as shown in Fig. 3b. Thus, by Figs. 3a and b, DIST achieves a high reachability, but faces several broken link situations leading to a low PDR. Otherwise, ACDB reaches a PDR slightly higher



(a) Reachability (b) PDR (c) Average Delay

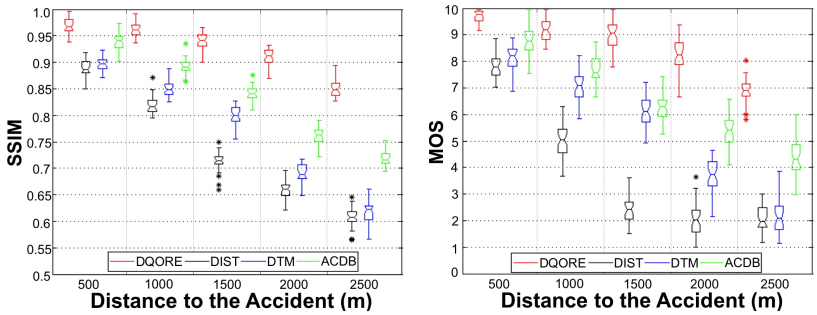
Figure 3. QoS-based metrics for all protocols with 200 veh/km²

in comparison to DTM, i.e., around 4.3%. This is because sometimes DTM elects farthest relay nodes such as DIST, mainly when there are few neighboring vehicles. Further, ACDB adapts FF_t depending on the number of neighbors, thus, increasing PDR.

The impact of the DBF and PMF phases are significant on the delivery delay over the transmission (Fig. 3c), since PMF allows a great reduction of the average delay by using a contention-free forwarding. Here, we consider the average delay required by a $W(VF)$ to be transmitted in a given range starting from VS. DIST at DBF phase experiences the lower delay, due to the reduced number of hops achieved by the DB method. After, DIST is followed by ACDB, since this protocol reduces its $BackoffTimer$ when in presence of few nodes. Also, when DIST is compared with DQORE and DTM, the delay reduction provided by DIST is around 45.4% and 44.8%, respectively for 1000m, and 46.0% and 46.5% for 2000m of range. As mentioned previously, DQORE, according to its forwarding criteria, provides more effort to deliver flows with better quality, this could mean forwarding streams to alternative nodes, increasing transmission durations. Otherwise, DIST just tries to minimize the number of hops and does not make any effort. This might result in longer delays and path lengths for DQORE. However, the achieved delay levels are negligible even in video applications and are significantly lower than the requirements of 4 to 5 s defined by CISCO [Hatting et al. 2005].

As discussed before, QoS-based metrics (e.g., PDR) are not enough to measure the quality level from the user's perspective. Thus, aiming to understand and confirm the impact of the QoE-Indicators criterion, the results in Fig. 4 present SSIM and MOS. SSIM values range from 0 to 1, where a higher value means better video quality. In Fig. 4.a, DQORE keeps the SSIM values around 0.97 and 0.85. An average increase of 23.4%, 16.3%, and 10.8% compared to DIST, DTM, and ACDB, respectively. It presents more deeply results than those obtained in Fig. 3b and shows significant benefits to the user's experience. This occurred because DQORE can estimates when the quality of the transmitting flow decreases based on the different received frame types, codec configurations, and losses, allowing vehicles, through FF calculation, to switch to others nodes, before increasing damage on the flow quality. For instance, let's suppose a $W(VF_i)$ successfully received by a VR_i in $|W(VF_i)|$ ms. As the spatial distribution of vehicles does not change very quickly in a short period of time (e.g., 3 s), it is likely that VR_i, continue to receive successfully a greater number of packets until a new route becomes necessary. Thus, DQORE provides a trade-off between hop-length and video quality.

With respect to real-time video assessment, sometimes the correlation between the SSIM results and subjective scores does not have a high accuracy. In face of this, it



(a) Metric: SSIM; GoP size: 14 (b) Metric: MOS; GoP size: 20

Figure 4. QoE-based metrics for all protocols with 200 veh/km²

is fundamental to have MOS experiments in order to really understand the video quality level according to human perception [Aguiar et al. 2014]. The results in the Fig. 4.b reinforce the results in the Fig. 4.a. Besides that, these results demonstrate that the QoE-indicators metric can be successfully extended from the distortion model introduced by Shu Tao in [Tao et al. 2008] and performs well when employed in QORE. Fig. 4.b presents the average MOS scores for all protocols and confirms that DQORE allows the delivery of live video sequences with a good or excellent quality in urban VANETs.

5. Conclusion

This paper introduced QORE to real-time video dissemination with QoE-awareness in VANETs. QORE aims to share videos with a better quality than existing works, since it employs the Interleaving scheme and QoE-indicators criterion to support the selection of the best next hops, changing to other nodes as soon as lower quality is identified. Simulation results highlight the performance and QoE-awareness support of QORE by measuring the video quality levels when the distance to the sender varies. By creating backbones and according to its forwarding criteria, QORE provides a greater support to a self-organized dissemination of video flows with a higher quality from the user's point-of-view. This could mean forwarding of streams to alternative nodes by increasing transmission durations, but nonetheless, still are insignificant to real-time video requirements. In future works, we will perform a study with different video features (e.g., varying the GoP size and packetization), so that FF_t can be dynamically adjusted to allow better performance.

References

Aguiar, E., Riker, A., Cerqueira, E., Abelém, A., Mu, M., Braun, T., Curado, M., and Zeadally, S. (2014). A real-time video quality estimator for emerging wireless multimedia systems. *Wireless Networks*, pages 1–18.

Claypool, M. and Zhu, Y. (2003). Using interleaving to ameliorate the effects of packet loss in a video stream. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 508–513. IEEE.

Di Felice, M., Bedogni, L., and Bononi, L. (2013). Group communication on highways: An evaluation study of geocast protocols and applications. *Ad Hoc Networks*, 11(3):818–832.

Felice, M. D., Cerqueira, E., Melo, A., Gerla, M., Cuomo, F., and Baiocchi, A. (2014). A distributed beaconless routing protocol for real-time video dissemination in multimedia VANETs. *Computer Communications*, (1):1–13.

Gerla, M., Wu, C., Pau, G., and Zhu, X. (2014). Content distribution in VANETs. *Vehicular Communications*, 1(1):3–12.

Hatting, C. et al. (2005). *End-to-end qos network design*. Cisco Press.

Immich, R., Borges, P., Cerqueira, E., and Curado, M. (2014). Adaptive motion-aware fec-based mechanism to ensure video transmission. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–6. IEEE.

Kakkasageri, M. and Manvi, S. (2014). Information management in vehicular ad hoc networks: A review. *Journal of Network and Computer Applications*, 39:334–350.

Li, Y., Naeimipoor, F., and Boukerche, A. (2014). Video dissemination protocols in urban vehicular ad hoc network: A performance evaluation study. *IEEE WCNC'14 Track 3 (Mobile and Wireless Networks)(IEEE WCNC'14 Track 3: NET)*.

Mohammed, A., Ould-Khaoua, M., Mackenzie, L. M., and Abdulai, J.-D. (2009). Dynamic probabilistic counter-based broadcasting in mobile ad hoc networks. In *Adaptive Science & Technology, 2009. ICAST 2009. 2nd International Conference on*, pages 120–127. IEEE.

Quadros, C., Cerqueira, E., Santos, A., and Gerla, M. (2014). A multi-flow-driven mechanism to support live video streaming on VANETs. In *Computer Networks and Distributed Systems (SBRC), 2014 Brazilian Symposium on*, pages 468–476.

Rosário, D., Zhao, Z., Santos, A., Braun, T., and Cerqueira, E. (2014). A beaconless opportunistic routing based on a cross-layer approach for efficient video dissemination in mobile multimedia IoT applications. *Computer Communications*, 45:21–31.

Slavik, M. and Mahgoub, I. (2013). Spatial distribution and channel quality adaptive protocol for multihop wireless broadcast routing in VANET. *Mobile Computing, IEEE Transactions on*, 12(4):722–734.

Slavik, M., Mahgoub, I., and Alwakeel, M. M. (2014). Analysis and evaluation of distance-to-mean broadcast method for VANET. *Journal of King Saud University-Computer and Information Sciences*, 26(1):153–160.

Tao, S., Apostolopoulos, J., and Guérin, R. (2008). Real-time monitoring of video quality in IP networks. *IEEE/ACM Transactions on Networking (TON)*, 16(5):1052–1065.

Torres, A., Calafate, C. T., Cano, J.-C., Manzoni, P., and Ji, Y. (2014). Evaluation of flooding schemes for real-time video transmission in VANETs. *Ad Hoc Networks*, 24, Part B(0):3 – 20. Modeling and Performance Evaluation of Wireless Ad-Hoc Networks.

Video Sequences, N. R. L. (2014). Videos used in the MOS experiments ('truck accident') < URL https://www.youtube.com/channel/UCUlJSLvBpeJbLAr6IsVXbJA/videos>.

Wang, R., Almulla, M., Rezende, C., and Boukerche, A. (2014). Video streaming over vehicular networks by a multiple path solution with error correction. In *Communications (ICC), 2014 IEEE International Conference on*, pages 580–585. IEEE.

Yao, X.-W., Wang, W.-L., Yang, S.-H., Cen, Y.-F., Yao, X.-M., and Pan, T.-Q. (2014). Ipb-frame adaptive mapping mechanism for video transmission over ieee 802.11 e wlans. *ACM SIGCOMM Computer Communication Review*, 44(2):5–12.

Inferência de Desempenho: Uma Nova Abordagem para o Planejamento da Capacidade de Aplicações na Nuvem

Marcelo Gonçalves, Matheus Cunha, Américo Sampaio, Nabor C. Mendonça

[1]Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321, Edson Queiroz, CEP 60811-905 Fortaleza, CE

{marcelocg,mathcunha}@gmail.com,{americo.sampaio,nabor}@unifor.br

Resumo. *Este trabalho propõe uma nova abordagem para apoiar o planejamento da capacidade de aplicações em nuvens que oferecem infraestrutura-como-serviço (IaaS). A abordagem proposta tem como premissa a existência de uma relação de capacidade entre diferentes configurações de recursos de um dado provedor de nuvem IaaS, com a qual é possível prever (ou "inferir"), com alta precisão, o desempenho esperado de uma aplicação para certas configurações de recursos e cargas de trabalho, tendo com base o desempenho da aplicação observado para outras configurações de recursos e cargas de trabalho neste mesmo provedor. Resultados empíricos preliminares, obtidos a partir da avaliação do desempenho de uma popular aplicação de* blogging *(WordPress) em um provedor de nuvem público (Amazon EC2), mostram que a nova abordagem consegue reduzir significativamente (acima de 85%) o número total de cenários de implantação da aplicação que precisam de fato ser avaliados na nuvem.*

Abstract. *This work proposes a novel approach to support application capacity planning in infrastructure-as-a-service (IaaS) clouds. The proposed approach relies on the assumption that there exists a capacity relation between different resource configurations offered by a given IaaS cloud provider, enabling one to predict (or "infer"), with high accuracy, an application's expected performance for certain resource configurations and workloads, based upon its observed performance for other resource configurations and workloads in that same provider. Preliminary empirical results, obtained from evaluating the performance of a well-known blogging application (WordPress) in a public cloud provider (Amazon EC2), show that the proposed approach can significantly reduce (over 85%) the total number of application deployment scenarios that need to be effectively tested in the 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]. Parte desse desafio envolve tentar descobrir a melhor maneira de implantar a aplicação na nuvem, considerando os vários tipos de recursos (em particular, máquinas virtuais) oferecidos pelo provedor, sob a perspectiva de diferentes requisitos e critérios de qualidade [Gonçalves Junior et al. 2015].

Em geral, provedores de nuvens IaaS cobram seus usuários em função do tempo de utilização dos recursos solicitados, cujos preços variam conforme a capacidade (normalmente medida por características técnicas como quantidade de núcleos de processamento, tamanho de memória e espaço de armazenamento) de cada recurso. Dessa forma, para calcular o custo de operação de uma aplicação na nuvem, é preciso estimar ou medir como a aplicação responderá a diferentes níveis de demanda, em termos de indicadores de desempenho como tempo de resposta ou vazão, quando executada sob diferentes configurações e perfis de máquinas virtuais. Na prática, isso significa que cabe ao usuário da nuvem identificar, dentre as possíveis configurações de máquinas virtuais ofertadas por um ou mais provedores de nuvem, aquelas de menor custo capazes de executar a aplicação mantendo-se níveis satisfatórios para os indicadores de desempenho.

Um grande problema começa a se desenhar para o usuário da nuvem ao seguir essa abordagem: a fase de avaliação da aplicação pode atingir patamares elevados de tempo e custo, em razão das necessidades de variação da demanda, da arquitetura de implantação e das configurações de recursos utilizadas para hospedar cada camada da aplicação [Silva et al. 2013]. Ainda que certos provedores IaaS ofereçam descontos ou pacotes de horas grátis para novos clientes, em geral esses incentivos, por estarem limitados a máquinas de pequeno porte, são insuficientes para suportar a carga de uma aplicação real em produção. Assim, executar uma aplicação real, tipicamente implantada em arquitetura de várias camadas [Jayasinghe et al. 2011], em máquinas virtuais de tamanho considerável e por longos períodos de tempo, apenas para estudar o seu comportamento, pode se traduzir em um custo alto que dificulte ou até mesmo inviabilize o próprio projeto de migração dessa aplicação para a nuvem [Beserra et al. 2012].

Vários trabalhos já foram propostos com o intuito de apoiar o planejamento da capacidade de aplicações em nuvens IaaS. Em linhas gerais, esses trabalhos podem ser classificados de acordo com duas abordagens distintas quanto à estratégia de avaliação do desempenho da aplicação. Trabalhos que seguem a primeira abordagem, referenciada neste trabalho como *abordagem preditiva*, visam estimar ou simular o desempenho esperado da aplicação para determinadas configurações de recursos e determinados níveis de carga, sem necessariamente ter que implantá-la na nuvem [Malkowski et al. 2010, Li et al. 2010, Li et al. 2011, Fittkau et al. 2012, Jung et al. 2013]. Apesar do baixo custo oferecido aos usuários, que não precisam pagar por recursos de nuvem durante a fase de avaliação, esse trabalhos têm como maior limitação a ainda baixa precisão das técnicas de predição de desempenho, particularmente daquelas baseadas em simulação [Fittkau et al. 2012]. Já os trabalhos que fazem parte da segunda abordagem, aqui referenciada como *abordagem empírica*, têm como objetivo medir o desempenho real da aplicação através de sua efetiva implantação na nuvem e da realização de testes de carga [Jayasinghe et al. 2012, Silva et al. 2013, Cunha et al. 2013a, Scheuner et al. 2014]. Por executarem a aplicação no próprio ambiente de nuvem, esses trabalhos conseguem resultados significativamente mais precisos no que diz respeito à seleção das melhores configurações de recursos para cargas de trabalho específicas. No entanto, uma limitação importante desses trabalhos é a necessidade de se testar exaustivamente uma grande quantidade de configurações de recursos e cargas de trabalho, implicando em altos custos durante a fase de avaliação.

Este trabalho propõe uma nova maneira de apoiar os usuários de nuvens IaaS a

identificarem as melhores (i.e., mais baratas) configurações de recursos capazes de satisfazer as demandas específicas de suas aplicações. A nova abordagem tem como premissa a existência de uma relação de capacidade entre diferentes configurações de recursos oferecidas por um dado provedor de nuvem, com a qual é possível prever (ou "inferir"), com alta precisão, o desempenho esperado da aplicação para determinadas configurações de recursos. A predição ou inferência é realizada com base no desempenho observado da aplicação para outras configurações de recursos e cargas de trabalho no mesmo provedor. Por exemplo, se a aplicação atendeu satisfatoriamente a demanda para uma configuração de recursos de determinada capacidade sob uma determinada carga de trabalho, é muito provável que ela também vá atendê-la para outras configurações de maior capacidade sob a mesma carga de trabalho. Analogamente, se a aplicação não atendeu a demanda para uma determinada configuração de recursos sob uma determinada carga de trabalho, muito provavelmente ela também não irá atendê-la para a mesma configuração sob cargas de trabalho maiores. Através do uso de inferência, a abordagem permite avaliar uma ampla variedade de cenários de implantação da aplicação, sendo que apenas uma parte relativamente pequena desses cenários precisa de fato ser implantada e executada na nuvem. Dessa forma, a abordagem consegue obter o melhor das duas abordagens previamente citadas, produzindo resultados de alta precisão (característicos da abordagem empírica) mas com significativa redução de custo (característica da abordagem preditiva).

A próxima seção apresenta um novo processo de avaliação de capacidade para aplicações na nuvem, fundamentado no conceito de inferência de desempenho. A Seção 3 descreve os resultados de uma avaliação preliminar do novo processo envolvendo a implantação de uma aplicação real em um provedor de nuvem IaaS público. A Seção 4 compara o processo proposto com outros trabalhos relacionados. Por fim, a Seção 5 oferece algumas conclusões e sugestões para trabalhos futuros.

2. Processo de Avaliação de Capacidade por Inferência de Desempenho

2.1. Conceitos e Terminologia

Antes de apresentarmos o processo, é necessário definirmos alguns conceitos importantes relacionados ao domínio da avaliação da capacidade de aplicações na nuvem (ver Tabela 1). A definição desses conceitos também serve para estabelecer a terminologia que será utilizada na descrição do processo, feita a seguir.

2.2. Dados de Entrada

O principal dado de entrada esperado pelo processo é o valor de referência (ou SLO), o qual será usado para determinar se a aplicação atingiu os requisitos mínimos de desempenho exigidos em cada cenário de execução. Além do SLO, o processo precisa também conhecer quais são as cargas de trabalho sob as quais o desempenho da aplicação deverá ser avaliado. Outro dado importante que deve ser passado como entrada para o processo é o espaço de implantação da aplicação. Para isso, o processo deve ser alimentado com três parâmetros: (i) uma lista de tipos de máquinas virtuais fornecidos pelo provedor no qual deseja-se hospedar a aplicação; (ii) a quantidade máxima de máquinas virtuais de cada tipo que irá compor cada configuração a ser avaliada; e (iii) um ou mais critérios para estabelecimento das relações de capacidade entre as configurações do espaço de implantação. A Seção 3 ilustra alguns critérios que podem ser usados para este fim.

Tabela 1. Conceitos e terminologia utilizados no artigo.

Conceito	Definição
Aplicação sob teste	Um sistema computacional, possivelmente implementado em uma arquitetura multicamadas, para o qual se deseja observar o comportamento em um ambiente de computação em nuvem e ao qual estão associadas uma ou mais *métricas de desempenho*.
Métrica de desempenho	Uma característica ou comportamento mensurável de forma automatizada e comparável a um *valor de referência*, capaz de indicar o grau de sucesso de uma execução da aplicação sob teste. É dependente do domínio da aplicação. Ex.: tempo de resposta, quadros por segundo.
Valor de referência	Um valor predefinido como minimamente aceitável para uma métrica de desempenho após uma execução da aplicação sob teste. Este valor, também referenciado neste trabalho como SLO (*Service Level Objective*), serve como base de comparação para que se classifique a aplicação como capaz de ser executada em uma certa *configuração* de máquinas virtuais e sob uma certa *carga de trabalho*.
Carga de trabalho	Representa o tamanho da demanda que será imposta à aplicação sob teste em uma execução. Sua unidade de medida é dependente do domínio da aplicação. Ex.: tamanho dos arquivos de entrada para uma aplicação de compactação de arquivos, quantidade de usuários concorrentes para uma aplicação web, etc.
Tipos de máquinas virtuais	Classificam as máquinas virtuais fornecidas por um provedor conforme suas características técnicas (e.g., núcleos de processamento, tamanho de memória, espaço em disco), permitindo que o provedor de nuvem mantenha uma linha de produtos discreta e finita.
Categorias de máquinas virtuais	Agrupam os tipos de máquinas virtuais de um provedor de acordo com suas características técnicas, plataforma e/ou arquitetura de hardware e a natureza do uso a que se destinam. Ex.: categorias que priorizam consumo de memória, acesso a disco, processamento gráfico, etc.
Configuração	Um conjunto de máquinas virtuais de um mesmo tipo e, portanto, de uma mesma categoria. *Configurações* são usadas para implantar uma ou mais camadas arquiteturais (ex.: apresentação, negócio, persistência) da aplicação sob teste.
Espaço de implantação	Denota um conjunto limitado de configurações de máquina virtuais nas quais a aplicação sob teste será implantada e executada durante uma sessão de avaliação.
Relações de capacidade	Relativizam o poder computacional das diversas configurações que compõem o espaço de implantação. As *relações de capacidade* definem um grafo orientado sobre o espaço de implantação onde os vértices correspondem às configurações e as arestas indicam a superioridade ou inferioridade (dependendo da direção da aresta) de uma configuração em relação a outra em termos de poder computacional.
Níveis de capacidade	Estabelecem uma hierarquia sobre as relações de capacidade definidas entre as configurações do espaço de implantação. Nessa hierarquia, configurações classificadas em um mesmo nível de capacidade seriam equivalentes (ou indistinguíveis) em termos de poder computacional.

2.3. Atividades

As principais atividades executadas pelo processo de avaliação de capacidade são ilustradas no diagrama da Figura 1. Nesse diagrama, atividades destacadas com o rótulo ≪A≫ são abstratas, devendo ser customizadas pelos usuários do processo de acordo com diferentes estratégias de avaliação (descritas na Seção 2.4). As demais atividades são concretas, sendo executadas independentemente da aplicação sob teste ou da estratégia de avaliação utilizada.

A execução do processo acontece de forma cíclica, com as atividades agrupadas em quatro fases distintas: (i) seleção do cenário de execução da aplicação; (ii) execução da aplicação; (iii) inferência de desempenho; e (iv) seleção do próximo cenário de execução. Cada uma dessas fases será detalhada a seguir.

2.3.1. Seleção do cenário de execução

A primeira atividade dessa fase é a escolha de uma carga de trabalho. Essa é uma atividade abstrata, significando que diferentes estratégias podem ser empregadas nessa escolha, por exemplo, selecionando um carga de trabalho maior ou menor dentre aquelas fornecidas como dados de entrada ao processo. Depois de selecionar a carga inicial, o processo seleciona uma categoria de máquinas virtuais. No caso da categoria, a ordem ou método utilizado na escolha é irrelevante para o processo, uma vez que todas as catego-

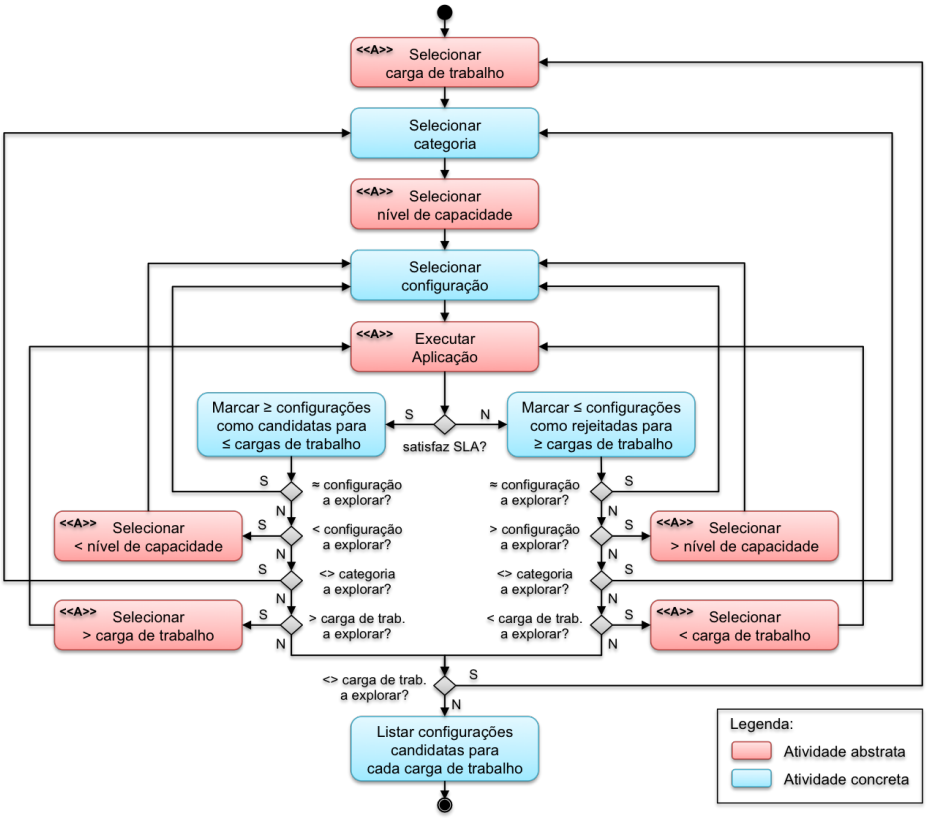


Figura 1. Diagrama de atividades do processo de avaliação de capacidade.

rias do espaço de implantação deverão ser avaliadas. Em seguida, o processo seleciona um nível de capacidade dentre aqueles presentes no espaço de implantação. Essa também é uma atividade abstrata, uma vez que níveis de capacidade mais acima ou mais abaixo na hierarquia podem ser escolhidos, a depender da estratégia de avaliação utilizada. Por fim, o processo seleciona uma configuração do nível de capacidade previamente selecionado. A ordem de seleção das configurações também é irrelevante, uma vez que todas as configurações daquele nível de capacidade devem ser avaliadas.

2.3.2. Execução da aplicação

Uma vez escolhidos uma carga de trabalho, uma categoria, um nível de capacidade e uma configuração, o processo está apto a executar a aplicação na nuvem. A execução da aplicação também é uma atividade abstrata do processo, pois depende de uma série de fatores que são específicos de cada aplicação ou plataforma de nuvem, como as tecnologias necessárias parar implantar os componentes da aplicação na nuvem bem como para submetê-los aos níveis de carga de trabalho desejados. Após a execução da aplicação, o processo analisa o resultado obtido e passa para a fase de inferência de desempenho.

2.3.3. Inferência de desempenho

Nesta fase, o processo se bifurca, atingindo seu primeiro ponto de decisão. A partir da análise do resultado da execução, que é feita comparando-se os indicadores obtidos para a

métrica de desempenho utilizada frente ao valor de referência (SLO) desejado, o processo determina se a aplicação é ou não capaz de atender à demanda imposta sobre ela com a atual configuração. Se a aplicação satisfaz o SLO, o processo assinala a configuração atual como uma *configuração candidata* para o atual nível de carga. Do contrário, o processo assinala a configuração atual como uma *configuração rejeitada* para esse nível de carga.

É neste momento que a abordagem de inferência de desempenho, proposta originalmente neste trabalho, entra em ação. Com base nas relações de capacidade presentes no espaço de implantação, o processo pode "inferir" o provável desempenho da aplicação para outras configurações e cargas de trabalho ainda não avaliadas. Se o processo identificou que uma certa configuração consegue satisfazer a demanda imposta à aplicação sob uma certa carga de trabalho, intuitivamente qualquer outra configuração de maior poder computacional também será capaz de fazê-lo sob a mesma carga de trabalho. Similarmente, é intuitivo concluir que a mesma configuração também será capaz de satisfazer o SLO da aplicação sob cargas de trabalho menores. Assim, usando as informações sobre as relações de capacidade existentes entre as configurações do espaço de implantação, o processo também assinala como candidatas para o atual nível de carga todas as outras configurações identificadas como sendo de "maior capacidade" que a configuração atual de acordo com o espaço de implantação. Da mesma forma, o processo também assinala a configuração atual como candidata para todos os níveis de carga inferiores ao nível de carga atual.

O caso em que a configuração atual não satisfaz o SLO da aplicação é tratado de modo análogo. Nesse caso, o processo assinala como rejeitadas para o atual nível de carga todas as outras configurações identificadas como sendo de "menor capacidade" que a configuração atual de acordo com o espaço de implantação. O mesmo acontece com a configuração atual, que também é assinalada como rejeitada para todos os outros níveis de carga superiores ao nível de carga atual.

2.3.4. Seleção do próximo cenário

Após a fase de inferência de desempenho, o processo seleciona os elementos que comporão o próximo cenário de execução a ser avaliado, ou encerra sua execução, caso não haja mais cenários a explorar. Nesse caso, o processo produz, como saída, uma lista contendo todas as configurações assinaladas como candidatas para cada carga de trabalho avaliada, em ordem crescente de preço.

A seleção do próximo cenário inclui a escolha de uma nova configuração do atual nível de capacidade, a escolha de um novo nível de capacidade (maior ou menor que o nível de capacidade atual), a escolha de uma nova categoria, ou a escolha de uma nova carga de trabalho (maior ou menor que o nível de carga atual). As escolha de um novo nível de capacidade ou de uma nova carga de trabalho vai depender do resultado da execução da aplicação no cenário atual, na medida em que o processo irá tentar diminuir (aumentar) o poder computacional da configuração atual ou, alternativamente, aumentar (diminuir) o nível de carga atual, caso a aplicação tenha alcançado (ou não) o SLO desejado. Por essa razão, essas escolhas também são consideradas atividades abstratas, a serem definidas como parte da customização do processo com diferentes estratégias de avaliação.

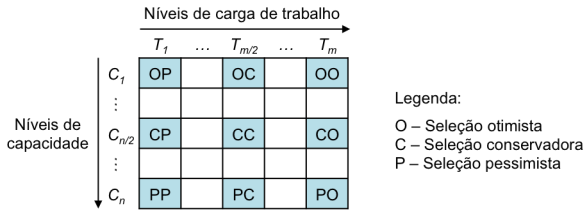


Figura 2. Heurísticas para seleção de configurações e cargas de trabalho.

2.4. Estratégias de Avaliação

Conforme mencionado anteriormente, todas as atividades abstratas do processo (com exceção da atividade de execução da aplicação na nuvem) devem ser customizadas de acordo com diferentes estratégias de avaliação. Essas atividades incluem, basicamente, a escolha de cargas de trabalho e níveis de capacidade. Tais escolhas influenciam diretamente a maneira através da qual o processo explora o espaço de implantação, tendo um forte impacto no alcance da inferência de desempenho.

Como exemplo, considere o caso de um espaço de implantação onde nenhuma configuração é capaz de atender a demanda da aplicação sob qualquer nível de carga. Nesse caso, iniciar o processo de avaliação pelas configurações do nível de capacidade mais baixo sob cargas de trabalho maiores não seria uma boa estratégia, uma vez que o número de configurações e cargas de trabalho para os quais o desempenho esperado da aplicação poderia ser inferido seria muito pequeno. Por outro lado, iniciar o processo pelas configurações de nível de capacidade mais alto sob cargas de trabalho menores seria um estratégia muito melhor, já que assim seria possível inferir o desempenho da aplicação para praticamente todas as outras configurações e todas as outras cargas de trabalho, representando uma grande economia de tempo e custo.

Esses dois extremos ilustram bem o desafio de se escolher os cenários de execução mais promissores do ponto de vista da inferência de desempenho. A fim de enfrentar esse desafio, este trabalho introduz o conceito das *heurísticas de seleção*, que agregam táticas a serem observadas no momento em que o processo, via alguma estratégia de avaliação, precisa escolher uma nova configuração ou uma nova carga de trabalho para compor um novo cenário de execução. Nesse sentido, foi inicialmente definido um conjunto de três táticas de seleção, denominadas *otimista*, *conservadora* e *pessimista*, respectivamente, aplicáveis tanto à escolha de novas cargas de trabalho quanto à escolha de novos níveis de capacidade. A combinação dessas três táticas na escolha de novos cenários de execução dá origem a nove heurísticas de seleção, ilustradas na Figura 2.

Nessa figura, as heurísticas são identificadas por diferentes pares de letras posicionados ao longo da matriz que representa o espaço de implantação. A primeira letra que identifica a heurística refere-se à tática usada na escolha da configuração (linha), enquanto a segunda letra refere-se à tática usada na escolha da carga de trabalho (coluna). Como pode-se observar, a tática otimista leva à escolha de configurações menores e cargas de trabalho maiores. Já a tática conservadora leva à escolha de configurações e cargas de trabalho de nível intermediário. Por fim, a tática pessimista leva à escolha de configurações maiores e cargas de trabalho menores. Cada heurística é aplicada recursivamente, de modo a explorar subconjuntos cada vez menores do espaço de implantação a cada nova iteração do processo. Nesse contexto, os termos menores, intermediários e maiores são

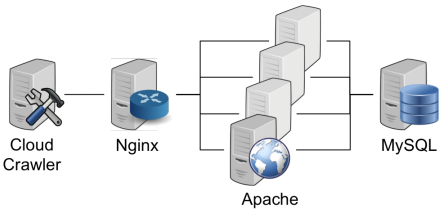


Figura 3. Arquitetura de implantação e avaliação do WordPress na Amazon EC2.

relativos, significando os elementos menores, intermediários e maiores, respectivamente, dentre aqueles ainda não explorados no espaço de implantação.

3. Avaliação Experimental

Esta seção descreve o experimento realizado como forma de verificação do processo de avaliação de capacidade apresentado anteriormente. Inicialmente, é apresentada a metodologia utilizada para a condução do experimento. Em seguida, são apresentados os resultados obtidos por cada uma das nove heurísticas de seleção propostas. Esses resultados são usados tanto para uma comparação qualitativa das heurísticas entre si, quanto para atestar a eficiência do processo proposto e de sua abordagem de inferência de desempenho.

É importante mencionar que o processo proposto foi implementado e está disponível na forma de uma ferramenta web,[1] a qual foi utilizada para executar o experimento descrito a seguir. Devido a restrições de espaço, os detalhes da implementação do processo bem como de sua ferramenta de apoio estão fora do escopo deste artigo.

3.1. Metodologia

O experimento consistiu na realização de sessões de avaliação de capacidade de uma aplicação web real (WordPress,[2] escolhida por ser uma das aplicações de criação e administração de *blogs* mais utilizadas atualmente) implantada em um provedor de nuvem também real (Amazon EC2,[3] escolhido por ser o líder de mercado entre provedores IaaS públicos). O WordPress foi implantado em duas camadas: uma para o banco de dados MySQL, e outra para o servidor de aplicação, executada pelo servidor Apache HTTPD. Como balanceador de carga, foi utilizada uma máquina dedicada executando o servidor web Nginx.

Devido a restrições de custo e tempo, o experimento limitou-se a variar apenas a camada de aplicação, usando de 1 a 4 servidores Apache executando o WordPress. A execução dos testes foi orquestrada pelo ambiente Cloud Crawler [Cunha et al. 2013b, Cunha et al. 2013a], que automatizou as tarefas de iniciar e parar todas as instâncias de máquinas virtuais, configurar o balanceador de carga de acordo com o número de instâncias testadas na camada de aplicação, iniciar e parar a execução dos testes, gerar as cargas de trabalho impostas à aplicação e, finalmente, coletar os dados de desempenho obtidos em cada teste. A Figura 3 ilustra a arquitetura utilizada para implantação e avaliação do WordPress na nuvem da Amazon.

[1] `http://cloud-capacitor.herokuapp.com/`.
[2] `https://wordpress.org/`.
[3] `http://aws.amazon.com/ec2`.

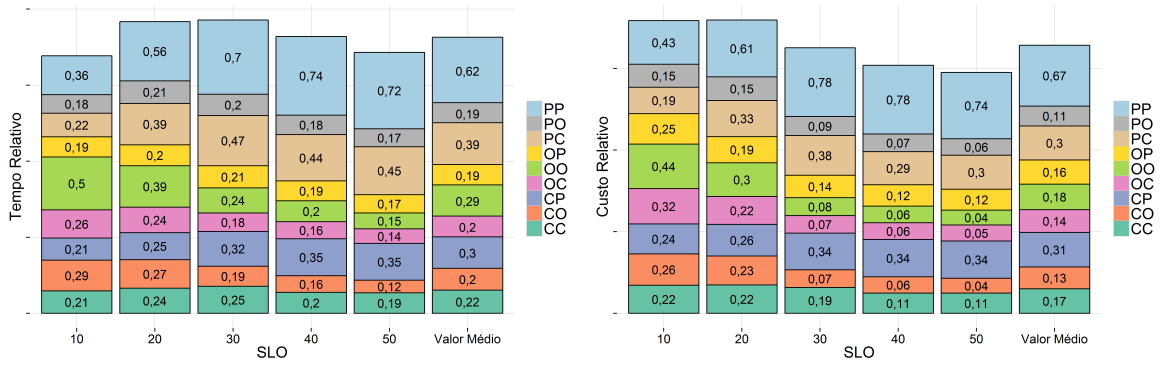
Para compor o espaço de implantação utilizado no experimento, foram escolhidos sete tipos de máquinas virtuais oferecidos pelo provedor Amazon EC2: *m3_medium*, *m3_large*, *m3_xlarge*, *m3_2xlarge*, *c3_large*, *c3_xlarge* e *c3_2xlarge*. Para cada um desses tipos, foram criadas configurações com 1, 2, 3 e 4 instâncias, levando a um total de 28 configurações diferentes no espaço de implantação, divididas em duas categorias distintas, "m3" e "c3". As relações de capacidade entre essas configurações foram definidas separadamente, para cada categoria, de modo a refletir o tipo e a quantidade de máquinas virtuais presentes em cada configuração. Assim, configurações com um certo número de máquinas virtuais de um determinado tipo eram consideradas de capacidade superior (inferior) a outras configurações contendo máquinas do mesmo tipo em menor (maior) quantidade. De maneira similar, configurações contendo um certo número de máquinas virtuais de um certo tipo eram consideradas de capacidade superior (inferior) a outras configurações com a mesma quantidade de máquinas mas de tipos diferentes se estes tipos fossem inferiores (superiores) ao tipo da primeira configuração, de acordo com a classificação dos tipos definidas pelo próprio provedor de nuvem. Por exemplo, a configuração composta por 3 máquinas to tipo *m3_2xlarge* era considerada superior a outra configuração composta por apenas 2 máquinas deste mesmo tipo. Da mesma forma, a configuração formada por 2 máquinas do tipo *c3_large* era considerada inferior a outra configuração com a mesma quantidade de máquinas do tipo *c3_xlarge*.

As cargas de trabalho utilizadas no experimento foram quantificadas em número de usuários concorrentes enviando requisições ao WordPress. Foi definido um total de 10 cargas de trabalho, representando 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000 usuários concorrentes, respectivamente.

De forma a estabelecer uma *baseline* para comparação da eficiência e da acurácia do processo proposto, especificamente de suas diferentes heurísticas de seleção, foram coletados dados de desempenho do WordPress na nuvem para cada um dos 280 cenários possíveis, ou seja, foram efetivamente realizados testes de desempenho da aplicação para cada uma das 28 configurações criadas sob cada uma das 10 cargas de trabalho especificadas. Esse conjunto de dados de execuções reais da aplicação foi denominado *oráculo*, e a estratégia necessária para gerar todos esses dados foi denominada heurística *Força Bruta* (em Inglês, *Brute Force – BF*). As nove heurísticas propostas foram então comparadas entre si e com a heurística BF.

Cada teste de desempenho consistiu em executar o WordPress utilizando uma das 28 configurações definidas para o espaço de implantação e então submetê-lo a uma das 10 cargas de trabalho especificadas durante um período de 1 hora. Durante os testes, um gerador de carga criava a quantidade de usuários corresponde à carga de trabalho sendo avaliada. Cada usuário realizava a seguinte sequência de requisições à aplicação: efetuar *logon*; inserir uma nova postagem; consultar a nova postagem; alterar a nova postagem; consultar postagens existentes por palavra-chave; alterar uma postagem existente; e, finalmente, efetuar *logoff*.

A métrica de desempenho utilizada no experimento foi o *tempo de resposta total*, ou seja, o tempo total decorrido entre o envio da primeira requisição da sequência acima e o momento em que o usuário recebeu a resposta para última requisição da sequência. Assim, para ser considerada como candidata para uma determinada carga de trabalho, uma configuração devia ser capaz de atender, sem erros, pelo menos 90% das sequências



(a) (b)

Figura 4. Eficiência das nove heurísticas de seleção em relação à heurística BF: (a) tempo de execução; (b) custo.

de requisições recebidas dos usuários da aplicação em um tempo total igual ou inferior ao valor do SLO, tal como definido no respectivo parâmetro de entrada do processo.

3.2. Resultados

3.2.1. Eficiência

Esta subseção apresenta os resultados de eficiência atingidos pelas heurísticas de seleção considerando-se duas métricas: *tempo de execução relativo* e *custo relativo*. Uma vez que a duração dos testes é igual em cada cenário, o tempo de execução relativo de uma determinada heurística é dado pela razão entre o número de vezes que a heurística executa a aplicação, e o número total de execuções da aplicação com a heurística BF. O custo relativo da heurística de seleção, por sua vez, é calculado pela razão entre a soma do custo de cada configuração efetivamente testada com essa heurística, e a soma dos custos de todas as configurações testadas com a heurística BF. Devemos notar que o custo de uma dada configuração depende do valor e da quantidade de máquinas virtuais que a compõem. Dessa forma, uma vez que os provedores podem fixar valores distintos para diferentes tipos de máquinas virtuais, o custo relativo de uma dada heurística de seleção será bastante influenciado pelas configurações específicas que a heurística selecionar para avaliar na nuvem.

A Figura 4 mostra os resultados para as duas métricas selecionadas, considerando os cinco SLOs investigados. Os resultados para a métrica tempo de execução relativo (Figura 4(a)) mostram que, sob SLOs mais brandos (ex: 50 segundos), as melhores heurísticas de seleção são OC e CO, oferecendo ganhos de 86% e 88%, respectivamente, com relação à BF. Porém, sob SLOs mais rígidos (ex: 10 segundos), as melhores heurísticas são PO e OP, com ganhos de 82% e 81%, respectivamente, com relação à BF. De fato, PO e OP, juntamente com CC, são em geral as melhores heurísticas para essa métrica, uma vez que seus resultados permanecem estáveis nos cinco SLOs, como indicado pelos valores médios (representados na coluna mais à direita dos dois gráficos). Os menores ganhos para essa métrica são obtidos com PP e PC, cujos ganhos, em média, podem chegar respectivamente a 38% e 61% com relação à BF.

No que diz respeito à métrica do custo relativo, uma análise da Figura 4(b) mostra que sob SLOs mais brandos os melhores resultados são obtidos com as heurísticas OO e

Tabela 2. Acurácia das heurísticas de seleção.

	SLO														
	10			20			30			40			50		
Heurística	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
CC	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,98	0,99	1,00	1,00	1,00	1,00	1,00	1,00
CO	1,00	1,00	1,00	1,00	1,00	1,00	0,99	1,00	0,99	1,00	1,00	1,00	1,00	1,00	1,00
CP	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,98	0,99	1,00	1,00	1,00	1,00	1,00	1,00
OC	1,00	1,00	1,00	1,00	1,00	1,00	0,99	0,99	0,99	1,00	1,00	1,00	1,00	1,00	1,00
OO	1,00	1,00	1,00	1,00	1,00	1,00	0,99	1,00	0,99	1,00	1,00	1,00	1,00	1,00	1,00
OP	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,98	0,99	1,00	1,00	1,00	1,00	1,00	1,00
PC	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,98	0,99	1,00	1,00	1,00	1,00	1,00	1,00
PO	1,00	1,00	1,00	1,00	1,00	1,00	0,99	1,00	0,99	1,00	1,00	1,00	1,00	1,00	1,00
PP	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,98	0,99	1,00	1,00	1,00	1,00	1,00	1,00

CO, com ambas oferecendo ganhos de até 96% em comparação à BF. No entanto, essas duas heurísticas não têm um bom desempenho sob SLOs mais rígidos; nesse caso, os melhores resultados são obtidos com as heurísticas PO, PC e CC, com ganhos entre 78% e 85% com relação à BF. Em geral, as melhores heurísticas para esta métrica são PO, OC e CO, oferecendo ganhos médios entre 86% e 89% com relação à BF. Os menores ganhos médios são oferecidos pelas heurísticas PP, PC e CP, sendo que PP mais uma vez se destaca com o pior resultado entre todas as nove heurísticas avaliadas.

Uma análise abrangendo os resultados de ambas as métricas revela que, no geral, as melhores heurísticas são PO, OP e CC, todas oferecendo ganhos de ao menos 75% com relação à heurística BF em todos os cinco SLOs.

3.2.2. Acurácia

Para medir a acurácia do processo de avaliação de capacidade, foram calculados os valores médios de *Precision*, *Recall* e *F-Measure* [Baeza-Yates and Ribeiro-Neto 1999] para os resultados produzidos por cada uma das heurísticas de seleção sob os diferentes valores de SLO avaliados, tomando como base os dados do oráculo. Para isso, os dados do oráculo foram utilizados para determinar se as configurações identificadas como candidatas (resultados positivos) e rejeitadas (resultados negativos) por cada heurística para uma determinada carga de trabalho eram de fato verdadeiras (nesse caso, as predições teriam sido corretas) ou falsas (nesse caso, as predições teriam sido erradas).

Os valores dessas três métricas para uma carga de trabalho i, denotados por P_i, R_i e F_i, respectivamente, são dados pelas seguintes fórmulas:

$$P_i = \frac{\textit{no. resultados positivos verdadeiros}}{\textit{no. resultados positivos verdadeiros} + \textit{no. resultados positivos falsos}}$$

$$R_i = \frac{\textit{no. resultados positivos verdadeiros}}{\textit{no. resultados positivos verdadeiros} + \textit{no. resultados negativos falsos}}$$

$$F_i = \frac{2 \times P_i \times R_i}{P_i + R_i}$$

A Tabela 2 mostra os valores médios de P, R e F, considerando as 10 cargas de trabalho, calculados para cada heurística de seleção sob os cinco níveis de SLO. Nota-se que em apenas um dos cinco SLOs o processo deixou de obter 100% de acurácia nas predições, apresentando uma taxa de erro inferior a 3% para os valores de *Precision* e *Recall*, e de aproximadamente 1% para os valores de *F-Measure*, que estabelece uma média ponderada entre as duas primeiras métricas [Baeza-Yates and Ribeiro-Neto 1999].

Uma investigação mais minuciosa dos dados de desempenho da aplicação na nuvem revelou que essa pequena perda na qualidade das predições foi devida a flutuações ocasionais no desempenho de alguns dos tipos de máquinas virtuais disponibilizadas pelo provedor. Essas flutuações levaram algumas das configurações avaliadas a terem um desempenho superior ao de outras configurações consideradas de maior capacidade de acordo com o espaço de implantação. Tais flutuações afetaram particularmente o desempenho da aplicação para o SLO de 30 segundos, refletindo em erros de predição. De fato, oscilações no desempenho da infraestrutura virtualizada oferecida por provedores de nuvem IaaS são relativamente comuns, como já observado em outros trabalhos [Iosup et al. 2011, Jayasinghe et al. 2011, Cunha et al. 2011]. Vale destacar que o impacto dessa instabilidade poderia ter sido mitigado, caso um número maior de execuções para cada par de configuração e demanda tivesse sido realizado. Mesmo assim, no contexto deste trabalho o nível de instabilidade observado foi muito baixo, afetando um único nível de SLO com taxa de erro médio de 1%. Esses resultados reforçam a nossa confiança de que a abordagem de inferência de desempenho tem potencial para atingir alta acurácia mesmo quando utilizada em aplicações e plataformas de nuvem reais.

4. Trabalhos Relacionados

Esta seção analisa várias soluções existentes para apoiar os usuários de nuvens IaaS no planejamento da capacidade necessária às suas aplicações. Conforme mencionado previamente, essas soluções seguem duas abordagens principais, aqui denominadas de preditiva e empírica.

As soluções da abordagem preditiva utilizam diferentes técnicas de predição do desempenho da aplicação, com destaque para a analogia com os resultados obtidos através da execução de diversos *benchmarks* na nuvem, normalmente coletados *a priori* pelo provedor da solução [Malkowski et al. 2010, Li et al. 2010, Jung et al. 2013]; simulação do comportamento esperado da aplicação através de um simulador de nuvem [Fittkau et al. 2012]; e reprodução na nuvem de eventos relevantes do ponto de vista de desempenho, como utilização de CPU, memória e disco, capturados a partir da execução local da aplicação [Li et al. 2011]. As abordagens que fazem analogia e simulação possuem a vantagem de serem de baixo custo, ao contrário da solução descrita em [Li et al. 2011], que necessita adquirir recursos da nuvem para reproduzir os eventos da aplicação. No entanto, todos esses trabalhos ainda deixam a desejar em termos de acurácia, devido a limitações importantes das técnicas de predição adotadas. Mais especificamente, a predição por analogia tem pouca eficácia se os *bechmarks* disponíveis não possuem perfis de comportamento similares ao da aplicação sob teste. Já os simuladores de nuvem ainda não conseguem atingir um nível de fidelidade próximo ao comportamento real de uma aplicação implantada em um provedor de nuvem público, chegando a apresentar diferenças de desempenho superiores a 30% [Fittkau et al. 2012]. Um problema similar ocorre com a solução que reproduz eventos da aplicação na nuvem, cujo mecanismo de captura de eventos ainda possui sérias limitações de ordem prática [Li et al. 2011].

As soluções empíricas, por outro lado, oferecem alta acurácia na avaliação do desempenho da aplicação na nuvem, uma vez que são baseadas em dados de desempenho obtidos diretamente no provedor [Jayasinghe et al. 2012, Silva et al. 2013, Cunha et al. 2013a, Scheuner et al. 2014]. Além disso, essas soluções são muito mais flexíveis, no sentido em que permitem aos usuários avaliar diferentes combinações de

componentes da aplicação sob as mais variadas configurações de recursos e cargas de trabalho. O ponto negativo das soluções que adotam a abordagem empírica é a necessidade de executar cada um dos cenários definidos pelo usuário, uma vez que elas não oferecem nenhum mecanismo voltado especificamente 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.

Existem outros trabalhos que adotam estratégias de planejamento de capacidade de curto prazo na nuvem, conhecidas como *auto-scaling* (e.g., [Morais et al. 2013]). Tais trabalhos visam ajustar dinamicamente os recursos da nuvem alocados à a aplicação, com base em regras de escalabilidade definidas pelo usuário e métricas coletadas a partir do monitoramento do comportamento da aplicação (ex: uso de CPU e memória). Alguns problemas relacionados com estas soluções é que nem sempre as regras especificadas pelos usuários levam em consideração a alocação das melhores configurações de máquinas virtuais (ex: em termos de custo e desempenho) para atender a demanda atual da aplicação.

Nesse contexto, o novo processo de avaliação de capacidade apresentado neste trabalho segue uma abordagem híbrida, combinando aspectos positivos das abordagens preditiva e empírica. Em contraste às soluções da abordagem preditiva, o novo processo realiza predições com base em relações de capacidade definidas entre diferentes configurações de recursos de um mesmo provedor de nuvem, e em resultados empíricos obtidos a partir da execução da própria aplicação neste provedor. Com isso, o novo processo consegue alta acurácia nas predições ao mesmo tempo em que reduz significativamente a quantidade de cenários de implantação que precisam ser efetivamente testados na nuvem. Além disso, acreditamos que o processo de inferência de desempenho também possa ser útil para apoiar soluções baseadas em auto-scaling, por exemplo, relevando as melhores configurações de máquinas virtuais para diferentes faixas de demanda da aplicação.

5. Conclusão e Trabalhos futuros

A tarefa de escolher adequadamente os recursos computacionais (ex.: 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 um novo processo de avaliação de capacidade por inferência de desempenho, que se mostrou uma solução ao mesmo tempo eficiente (em termos de custo e tempo) e eficaz (em termos da acurácia dos resultados) para apoiar o planejamento da capacidade de aplicações na nuvem.

Com relação aos trabalhos futuros, algumas possibilidades interessantes para melhoria ou extensão deste trabalho incluem: realizar novos experimentos visando investigar se os resultados reportados neste artigo são generalizáveis para outras aplicações e provedores de nuvem; investigar novas heurísticas de seleção de configurações e cargas de trabalho, que levem em conta dados sobre a utilização dos recursos da nuvem pela aplicação, como consumo de CPU e memória; e propor novos critérios para definir as relações de capacidade entre as diferentes configurações disponibilizadas pelo provedor de nuvem, por exemplo, considerando o custo de cada configuração, e investigar seu impacto no desempenho das heurísticas de seleção.

Referências

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

Beserra, P. V. et al. (2012). Cloudstep: A Step-by-Step Decision Process to Support Legacy Application Migration to the Cloud. In *IEEE MESOCA 2012*, pages 7–16.

Cunha, M. et al. (2011). Investigating the impact of deployment configuration and user demand on a social network application in the Amazon EC2 cloud. In *IEEE CloudCom 2011*, pages 746–751.

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*, pages 747–760.

Fittkau, F. et al. (2012). CDOSim: Simulating cloud deployment options for software migration support. In *IEEE MESOCA 2012*, pages 37–46.

Gonçalves Junior, R. et al. (2015). A Multi-Criteria Approach for Assessing Cloud Deployment Options Based on Non-Functional Requirements. In *ACM SAC 2015*.

Iosup, A. et al. (2011). On the performance variability of production cloud services. In *IEEE/ACM CCGrid 2011*, pages 104–113.

Jayasinghe, D. et al. (2011). Variations in performance and scalability when migrating n-tier applications to different clouds. In *IEEE CLOUD 2011*, pages 73–80.

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. (2010). CloudCmp: Comparing Public Cloud Providers. In *ACM SIGCOMM IMC 2010*, pages 1–14.

Li, A. et al. (2011). CloudProphet: Towards Application Performance Prediction in Cloud. In *ACM SIGCOMM 2011*, pages 426–427.

Malkowski, S. et al. (2010). CloudXplor: A tool for configuration planning in clouds based on empirical data. In *ACM SAC 2010*, pages 391–398.

Menascé, D. A. and Ngo, P. (2009). Understanding Cloud Computing: Experimentation and Capacity Planning. In *CMG 2009*.

Morais, F. J. A. et al. (2013). Autoflex: Service Agnostic Auto-scaling Framework for IaaS Deployment Models. In *IEEE/ACM CCGrid 2013*, pages 42–49.

Scheuner, J. et al. (2014). Cloud WorkBench – Infrastructure-as-Code Based Cloud Benchmarking. *arXiv preprint arXiv:1408.4565*.

Silva, M. et al. (2013). CloudBench: Experiment Automation for Cloud Environments. In *IEEE IC2E 2013*, pages 302–311.

VITreeM - Um Algoritmo baseado em Árvores para Alocação de Infraestruturas Virtuais

Ramon de Oliveira, Guilherme Piegas Koslovski

[1]Programa de Pós Graduação em Computação Aplicada (PPGCA)
Universidade do Estado de Santa Catarina - UDESC - Joinville, SC - Brasil

ramon.oliveira@joinville.udesc.br, guilherme.koslovski@udesc.br

Resumo. *As nuvens computacionais introduziram o provisionamento dinâmico de Infraestruturas Virtuais (IVs) disponibilizando computação, armazenamento e comunicação como serviços. Uma das questões investigadas nesse cenário é a alocação eficiente de recursos físicos para hospedar IVs. Por se tratar de um problema NP-Díficil, a busca por soluções ótimas possui um limite de aplicabilidade em cenários reais. Este artigo propõe um algoritmo para realizar essa tarefa, aplicando heurísticas baseadas em árvores para reduzir o espaço de busca. A proposta compreende uma estratégia para transformar grafos físicos e virtuais em árvores, que posteriormente, são simplificadas através da contabilização agrupada de informações. Essas inovações reduzem o número de comparações necessárias para identificar uma solução de alocação. Experimentos indicam que o algoritmo proposto encontra soluções em tempo viável para diversos cenários, mantendo uma alta taxa de aceitação e induzindo uma baixa fragmentação nos recursos físicos.*

Abstract. *Cloud Computing introduced dynamic provisioning of Virtual Infrastructures (VIs) delivering computing, storage and communication as services. One of the challenges investigated in this scenario is the efficient allocation of physical resources to host VIs. Due to its NP-Hard nature, the search for optimal solutions has a limited applicability in real scenarios. This paper proposes an algorithm to accomplish this task applying tree-based heuristics to reduce the search space. The proposal includes a strategy to transform physical and virtual graphs into trees, which subsequently are simplified by grouped accounting information. These innovations reduce the number of required comparisons to allocate a VI. Experiments indicate that the proposed algorithm finds solutions in feasible time for different scenarios while maintaining a high acceptance rate and the low physical infrastructure fragmentation.*

1. Introdução

A computação em nuvem introduziu a entrega de recursos (armazenamento, comunicação e processamento) virtualizados sob demanda, abstraindo a preocupação, por parte do usuário, sobre o gerenciamento, manutenção e localização dos recursos físicos [Zhang et al. 2010] [Mell and Grance 2011]. Especificamente, os provedores de *IaaS (Infrastructure as a Service*) ofertam máquinas virtuais (MVs), roteadores e redes virtuais como serviço, compondo as Infraestruturas Virtuais (IVs). Para que as IVs possam ser provisionadas, o provedor deve alocá-las [1] sobre os recursos físicos, isto

[1]Seguindo a literatura, os termos mapeamento e alocação possuem o mesmo significado neste artigo.

é, além de alocar as MVs é necessário mapear os roteadores e canais que compõem a IV [Fischer et al. 2013]. Formalmente, IVs e recursos físicos, são representados como grafos: os vértices denotam os roteadores e as MVs enquanto as arestas representam os canais de comunicação. Sob esse ângulo, o provedor deve encontrar um mapeamento do grafo da IV sobre o grafo físico. Essa tarefa pertence ao conjunto de problemas NP-Difícil, pois pode ser reduzida a outros cenários já provados serem deste conjunto [Chowdhury and Boutaba 2010].

A literatura especializada compreende trabalhos que exploram o problema de alocação de recursos buscando soluções ótimas ou aplicando heurísticas [Yu et al. 2008], [Chowdhury et al. 2009], [Cheng et al. 2011], [Alkmin et al. 2013], [Wang et al. 2013], [Cavalcanti et al. 2014]. Embora eficientes em seus domínios de aplicação, constata-se que os trabalhos, em sua maioria, solucionam apenas a alocação de redes virtuais (*VNE - Virtual Network Embedding*), ou seja, roteadores e canais de comunicação, não considerando a existência das MVs e, consequentemente, limitando o número de recursos participantes da formulação do problema. Ademais, a busca por soluções ótimas possui uma aplicabilidade limitada em diversos cenários. Como observado em [Luizelli et al. 2013], as propostas nem sempre utilizam topologias que representam cenários reais (físicos ou virtuais), afirmação que pode influenciar na taxa de aceitação das requisições.

Nesse contexto, este trabalho propõe um algoritmo heurístico possuindo como premissa a definição de que ambos os grafos (IV e físico) são representados na forma de árvores, ou seja, os grafos devem ser conexos e acíclicos. Essa premissa permite a diminuição do espaço de busca através da aplicação de estratégias para abstração das subárvores. Para permitir o uso do algoritmo com qualquer topologia física e virtual, é apresentada uma estratégia para transformação de grafos em árvores. Os resultados obtidos são avaliados considerando métricas como taxa de aceitação, tempo de mapeamento, carga e fragmentação do *datacenter*. Em suma, as principais contribuições deste trabalho são: i) a definição de um algoritmo para transformação de topologias (física ou IVs) em árvores; ii) um algoritmo para abstrair e simplificar a representação das capacidades requisitadas e residuais; iii) um algoritmo para alocação de IVs baseado em árvores.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 formaliza o problema de alocação de IVs enquanto a Seção 3 revisa os trabalhos relacionados. A Seção 4 descreve a solução proposta. A análise experimental é discutida na Seção 5. A Seção 6 apresenta as considerações finais e perspectivas para trabalhos futuros.

2. Alocação de Infraestruturas Virtuais

Através da virtualização, provedores IaaS podem abstrair os recursos disponíveis em *datacenters* para hospedar entidades virtuais isoladas, pertencentes a usuários distintos. Por exemplo, um único nó físico pode hospedar diversos nós virtuais (o mapeamento de nós virtuais para nós físicos é uma relação de n:1), enquanto canais virtuais podem se estender por vários canais da infraestrutura física (um caminho) [Chowdhury and Boutaba 2010]. Assim, um canal virtual que conecta dois nós virtuais u e v é mapeado sobre o caminho que conecta os nós físicos que hospedam os nós virtuais u e v (o mapeamento de canais virtuais para caminhos físicos é visto como um relacionamento n:m) [Fischer et al. 2013].

Em cenários típicos, existem restrições que devem ser respeitadas durante o mapeamento. Ou seja, a infraestrutura física deve ser capaz de suportar as capacidades soli-

citadas para os recursos virtuais. Por exemplo, os nós físicos devem suportar os requisitos mínimos de capacidade de processamento, memória e armazenamento exigidos pelo nó virtual. De forma semelhante, os canais físicos de comunicação devem suportar os requisitos de largura de banda, nível de confiança e qualidade de serviço exigidos pelo canal virtual [Alkmin et al. 2013] [Cavalcanti et al. 2014]. É sabido que a alocação não pode ser realizada aleatoriamente pois existem objetivos a serem alcançados. Ainda, a elasticidade de uma IV deve ser contemplada dinamicamente, ou seja, periodicamente um mecanismo deve verificar a necessidade de alteração dos recursos reservados. Usualmente, provedores buscam alocar o maior número de IVs de uma maneira eficiente, maximizando a taxa de aceitação e simultaneamente minimizando o número de recursos físicos compromissados. Ademais, as IVs podem ser posicionas em qualquer lugar, porém o mecanismo de alocação deve reduzir o espalhamento e a fragmentação dos recursos físicos, diminuindo o número de recursos ativos e os custos administrativos.

Formalmente, o problema de alocar IVs sobre a infraestrutura física pode ser descrito como uma extensão da classe de problemas VNE [Yu et al. 2008] [Lischka and Karl 2009] [Chowdhury et al. 2009]: $G = (N, E, C)$ denota uma infraestrutura física, na qual N representa o conjunto de nós físicos (servidores e roteadores), E o conjunto de canais físicos, e cada nó ou canal físico $a \in N \cup E$ está associado com o vetor de capacidade $C(a)$ indicando a quantidade de recursos que a oferece. De forma semelhante, $G_v = (N_v, E_v, C_v)$ denota uma requisição de IV, onde N_v e E_v representam o conjunto de nós e canais virtuais, respectivamente, e C_v é o vetor de capacidades de um nó ou canal virtual. Assim, o mapeamento de IVs sobre os recursos físicos é descrito como $M : G_v \rightarrow (M_N, M_P)$, onde $M_N : n^i \in N_v \rightarrow n \in N$ é um mapeamento de nó virtual para nó físico e $M_P : e^i \in E_v \rightarrow P$ é o mapeamento de um canal virtual sobre um conjunto P de canais físicos. Cada G_v é alocada considerando o estado atual de G (grafo residual). Por grafo residual entende-se o grafo com as capacidades dos nós e arestas reduzidas, de acordo com as alocações realizadas sobre estes. $M(G_v)$ é considerado um mapeamento válido se, para todo nó ou canal virtual de G_v, seu respectivo mapeamento não excede as capacidades em G, ou seja, $\forall n^i \in N_v, C_v(n^i) \leq C(M_N(n^i))$ e $\forall e^i \in E_v, C_v(e^i) \leq min(C(l), \forall l \in P)$.

3. Trabalhos Relacionados

A literatura atual compreende diversas soluções visando a alocação de recursos computacionais para hospedar entidades virtuais [Fischer et al. 2013]. As soluções existentes, em sua maioria, abordam uma subclasse do problema: *Virtual Network Embedding (VNE)*. Basicamente, VNE denota a alocação de um conjunto de roteadores virtuais (interconectados por enlaces virtuais) sobre topologias físicas [Chowdhury and Boutaba 2010].

Dentre as soluções revisadas, a abordagem de [Yu et al. 2008] introduziu a decomposição de um canal virtual sobre múltiplos caminhos físicos. Essa abordagem, combinada com uma reorganização periódica dos canais, alcançou um melhor aproveitamento da infraestrutura física. Inicialmente, a heurística buscava um posicionamento dos roteadores virtuais, e posteriormente, o mapeamento dos canais de comunicação. Embora o foco do trabalho não tenha compreendido a alocação de MVs, a métrica utilizada para identificação de candidatos potencialmente complexos era baseada no grau de conectividade do nó e serviu como base de comparação para o algoritmo proposto na Seção 4.

Por sua vez, [Lischka and Karl 2009] apresentou uma formulação baseada na identificação de grafos isomórficos, efetuando uma busca exaustiva pela presença de isomorfismo. Os resultados obtidos indicaram um elevado tempo de mapeamento (minutos). Em [Chowdhury et al. 2009] introduziu-se a alocação combinada dos canais de comunicação e roteadores virtuais, ou seja, ambos passaram a ser mapeados em um único passo pelo algoritmo. Essa inovação motivou a inclusão das MVs na formulação do problema e a concepção do algoritmo proposto no presente trabalho. Embora a diversidade das topologias virtuais não tenha sido o alvo dos trabalhos citados, os resultados obtidos motivaram o desenvolvimento da proposta de [Luizelli et al. 2013], que quantificou o impacto das topologias na taxa de aceitação das requisições. Seguindo essa linha, [Wang et al. 2013] propôs uma solução para topologias baseadas em árvores. Entretanto, a abordagem não considerava a alocação combinada de recursos de comunicação e processamento, limitando a representação de requisições e topologias físicas apenas na forma de árvores. Em contrapartida, [Cheng et al. 2011] e [Gong et al. 2014] não limitaram a formulação do problema para apenas uma topologia, explorando as particularidades de cada uma para identificação dos nós críticos (grau de conectividade).

Em [Alkmin et al. 2013], a formulação do VNE foi discutida considerando diversas informações para representação dos requisitos virtuais e capacidades dos recursos físicos. Recentemente, alguns trabalhos abordaram uma formulação que, além da alocação conjunta dos recursos de comunicação (roteadores e enlaces), introduziu a alocação de MVs [Cavalcanti et al. 2014] [Ruck et al. 2014]. Embora o presente trabalho tenha a alocação de IVs como alvo, alguns algoritmos e métricas oriundos de VNE foram considerados como diretrizes na formulação da solução: tempo de mapeamento aceitável, alta taxa de aceitação, baixa fragmentação da infraestrutura física, impacto das topologias de rede no processo de mapeamento e definição do nó de partida. Assim, a Seção 4 propõe uma abordagem para alocação de IVs aplicando conceitos referentes a árvores para diminuição do espaço de busca.

4. Uma Nova Abordagem para a Alocação de IVs

Este artigo propõe uma solução heurística (VITreeM - *Virtual Infrastructure Tree Mapping*) para a alocação *online* de IVs sobre uma infraestrutura física. VITreeM possui como premissa que ambos os grafos, de IV e infraestrutura física, estejam na forma de árvores. Em teoria dos grafos, uma árvore é um grafo não direcionado G que satisfaz as seguintes condições: *i)* ser conexo, ou seja, existe um caminho entre quaisquer dois vértices de G; e *ii)* não conter ciclos, ou seja, existe apenas um único caminho entre quaisquer dois vértices de G. Essa premissa é motivada pela utilização de árvores na composição de topologias de *datacenters* [Cisco 2007] [Niranjan Mysore et al. 2009] [Kandula et al. 2009] e pela possibilidade de aplicação de algoritmos polinomiais para alocação. Entretando, é sabido que as topologias de infraestruturas físicas e virtuais não necessariamente são especificadas como árvores, infringindo a condição de não existência de ciclos [Luizelli et al. 2013].

Dessa forma, VITreeM introduz uma estratégia para transformação de grafos em árvores abrangendo assim qualquer topologia. Em suma, a solução é decomposta em: *i)* transformação dos grafos em árvores conservando a representação das topologias originais; *ii)* definição dos nós de partida (as raízes das árvores) para alocação; *iii)* agrupamento em caixas para simplificar a representação dos recursos e diminuir o espaço de busca; *iv)* mapeamento das árvores guiado pelas definições da Seção 2.

4.1. Transformação de Grafos em Árvores

A estratégia para transformação de grafos em árvores é aplicada para as topologias físicas e virtuais, antes de efetuar o mapeamento. Como os grafos de entrada devem ser conexos, a infração nas propriedades de árvore ocorre na existência de ciclos entre os nós. Para remover os ciclos, a estratégia de transformação consiste em obter a árvore geradora máxima do grafo. Uma árvore geradora de um grafo $G = (N, E)$ é um subgrafo $T = (N^T, E^T)$ que possui as propriedades de uma árvore e contém todos nós de G, ou seja, $N^T = N$ e $E^T \subseteq E$. Em grafos ponderados a árvore geradora máxima de G é uma árvore geradora com o maior peso. Existe a possibilidade de haver mais de uma árvore geradora máxima, neste caso todas são equivalentes [Baase 2009]. Para obtenção da árvore geradora máxima de um grafo, utilizou-se uma adaptação do algoritmo de Kruskal, originalmente desenvolvido para obter a árvore geradora mínima [West et al. 2001]. Para tal, foi alterada a ordem de avaliação das arestas, substituindo a ordenação não-decrescente por uma ordenação não-crescente.

No grafo de uma infraestrutura física, as arestas que não estão presentes na árvore geradora máxima não serão consideradas no mapeamento. Porém, dentre todas árvores geradoras obtidas, a árvore geradora máxima é a que tem maiores chances de sucesso no mapeamento devido a maximização dos pesos nos canais de comunicação. A Figura 1 apresenta um exemplo de um grafo de infraestrutura física e a árvore geradora máxima da mesma. Nota-se que as arestas $(d, b), (d, c)$ foram removidas no processo de obtenção da árvore geradora máxima.

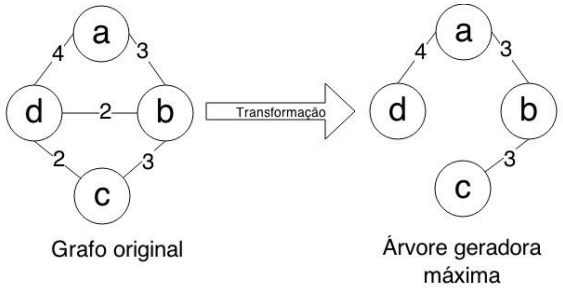


Figura 1. Definição de uma árvore geradora máxima para infraestruturas físicas.

No grafo de uma IV, $G_v = (N_v, E_v, C_v)$, descartar as arestas que não estão inclusas na árvore geradora máxima, $T_v = (N_v^T, E_v^T, C_v^T)$, não é uma opção aceitável, pois descaracteriza a estrutura da requisição. Para contornar este problema, o peso de toda aresta $(u, v) \in E_v - E_v^T$, é adicionado a toda aresta pertencente ao caminho entre u e v de T_v. A Figura 2 ilustra o processo de transformação aplicado a uma IV: a partir do grafo original obtém-se a árvore geradora máxima removendo as arestas $(d, b), (d, c) \in E_v$. Por fim, adiciona-se o peso destas arestas na árvore geradora máxima entre os nós de origem e destino. Por exemplo, para a aresta (d, b), seu caminho na árvore geradora máxima é *d-a-b*, logo, adiciona-se o peso da aresta $(d, b) \in E_v$ nas arestas $(d, a), (a, b) \in E_v^T$. Assim, a aresta $(d, b) \in E_v$ passa a ser representada pelo caminho *d-a-b* na árvore final, mantendo a conectividade entre os recursos virtuais conforme originalmente requisitado.

4.2. Nós de Partida para a Alocação

Por definição, o algoritmo precisa identificar um nó de cada árvore para iniciar a busca por uma solução. Conforme discutido em trabalhos anteriores [Yu et al. 2008]

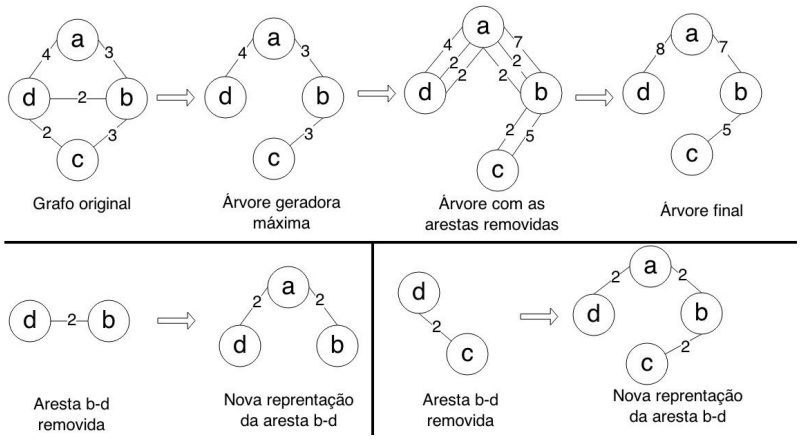
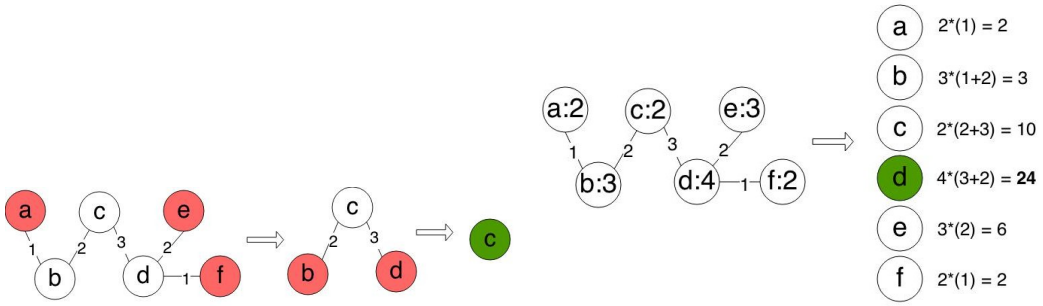


Figura 2. Processo de definição de uma árvore geradora máxima para uma IV.

[Gong et al. 2014], a definição do ponto de partida influencia na distribuição dos recursos virtuais sobre a infraestrutura física, pois ela define o modo como o algoritmo guiará a busca por recursos hospedeiros. Desta forma, a solução proposta utiliza duas estratégias para escolha dos nós iniciais: a identificação do centro da árvore e a busca pelo nó com a maior Capacidade Local de Recursos (*Local Resource Capacity* - LRC).

Em teoria dos grafos, a excentricidade de um nó n, pertencente a uma árvore, é definida como a maior distância do nó n para qualquer outro vértice. Por sua vez, o centro de uma árvore é um nó com a menor excentricidade e pode ser obtido através da remoção iterativa das folhas da árvore, restando apenas um ou dois nós [Knuth 1968]. A Figura 3(a) ilustra a obtenção do centro da árvore. As folhas (em vermelho) são removidas iterativamente até restar apenas o nó c que é o centro da árvore (em verde). A métrica LRC é utilizada para mensurar o potencial de mapeamento de um determinado nó, sendo calculada como o produto da capacidade do nó u e o somatório das capacidades das arestas diretamente conectadas ao nó ($LRC = C(u) * \sum C(u,v), \forall v \in N^T \wedge (u,v) \in E^T$) [Yu et al. 2008]. Quanto maior o valor do LRC para um nó, menor é a sua chance de ser mapeado devido a sua maior exigência de recursos. A Figura 3(b) ilustra o calculo de LRC para cada um dos nós presentes na árvore, escolhendo o nó d com maior LRC.



(a) Remoção iterativa de folhas para definição do nó central de uma árvore.

(b) Calculo do LRC [Yu et al. 2008].

Figura 3. Identificação dos nós de partida para alocação.

4.3. Agrupamento dos Recursos em Caixas de Capacidades

Um dos principais desafios identificados na alocação de IVs é a explosão combinatória resultante do número de candidatos disponíveis em *datacenters*. É sabido que a abstração dos candidatos através do agrupamento das informações reduz o número de operações necessárias para verificar se um mapeamento é factível [Ricci et al. 2003]. VITreeM explora uma heurística, chamada de caixas, para agrupar as informações nos ramos de uma árvore, diminuindo o espaço de busca. Através das caixas de capacidades, o algoritmo possui uma noção dos recursos disponíveis a partir de um determinado nó, sem a necessidade de uma investigação exaustiva na árvore. Uma caixa contém uma tupla com três valores de capacidades (canais de comunicação, processamento e roteamento) obtidos através do somatório das capacidades nas subárvores formadas a partir de um determinado nó (os valores de uma caixa são alterados dependendo da referência utilizada).

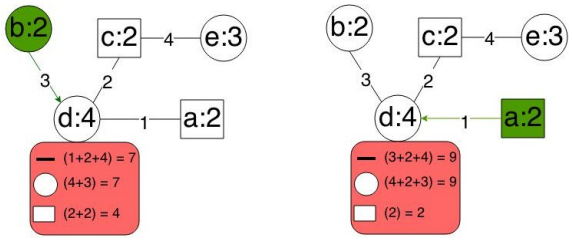


Figura 4. Exemplo de agrupamento de capacidades em caixas.

Na Figura 4 observa-se um exemplo do agrupamento em caixas de capacidades (em vermelho). Roteadores, servidores e canais de comunicação são representados por círculos, retângulos e linhas contínuas, respectivamente. A caixa do nó d, quando observada a partir do nó b ($caixa(b \rightarrow d)$) contém valores diferentes da caixa do nó d observada a partir do nó a ($caixa(a \rightarrow d)$), pois formam subárvores diferentes. O algoritmo utiliza essa abstração para ter uma noção da quantidade de recursos que estão presentes em uma subárvore, permitindo quantificar o seu potencial de mapeamento, sem a necessidade de explorar individualmente os elementos do espaço de busca.

4.4. Alocação de Infraestruturas Virtuais

O algoritmo VITreeM é o resultado da combinação das heurísticas anteriormente descritas, realizando as etapas na ordem em que foram apresentadas. Com foco na alocação *online* de requisições, este algoritmo recursivo aloca IVs individualmente em um determinado instante de tempo. O pseudocódigo apresentado no Algoritmo 1 descreve o funcionamento de VITreeM, recebendo como entrada: *i)* um grafo $G = (N, E, C)$ representando a infraestrutura física; *ii)* um grafo $G_v = (N_v, E_v, C_v)$ denotando a IV; *iii)* um nó central n_G da árvore T, sendo que $T = grafo\_para\_arvore(G)$; *iv)* o nó central n^v para a árvore T_v, sendo que $T_v = grafo\_para\_arvore(G_v)$; *v*) a árvore T; *vi*) a árvore T_v e *vii)* e um mapeamento $M = (M_N, M_P)$ inicialmente vazio. Se alguma solução for encontrada, o algoritmo fornece como resultado um mapeamento M de G_v sobre G indicando quais recursos físicos foram utilizados para hospedar a requisição de IV.

Algoritmo 1: $VITreeM$ - Alocação de IVs baseada em árvores.

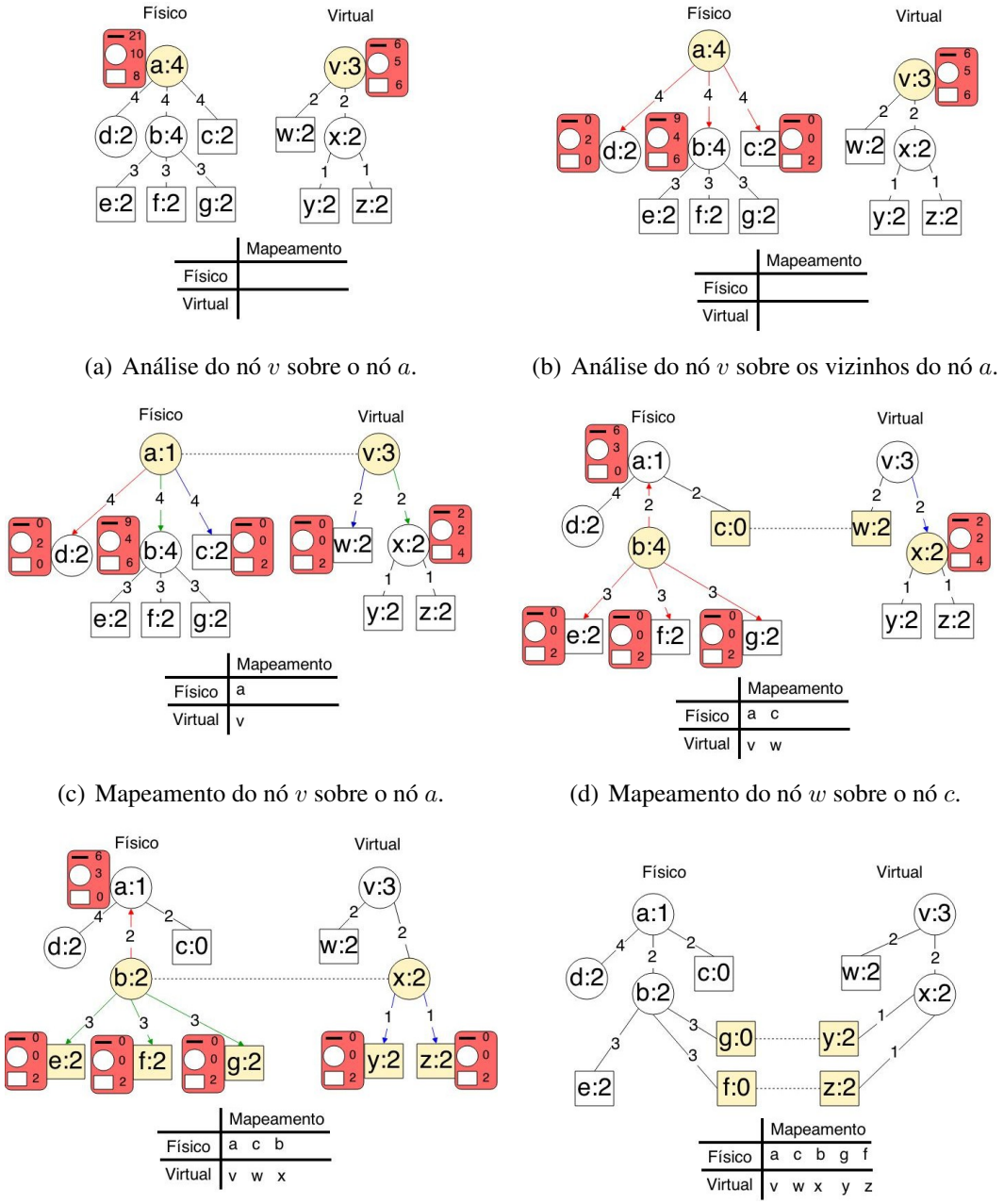
Data: $G = (N, E, C)$; $G_v = (N_v, E_v, C_v)$; $n_G \in T$; $n^v \in T_v$; T; T_v; $M = (M_N, M_P)$;
Result: mapeamento M de G_v sobre G, caso possível

1 **for** $\forall (u, n_G) \in T$ *tal que* $u \in T$ **do**
2 **if** $caixa(n_G \rightarrow u) \geq caixa(anterior(n^v) \rightarrow n^v)$ **then**
3 **if** $VITreeM(G, G_v, u, n^v, T, T_v, M)$ **then**
4 **return** (*verdadeiro*, M);
5 **if** $C(n_G) < C_v(n^v)$ **then**
6 **return** (*falso, null*);
7 $M_N(n^v) = n_G$;
8 $C(n_G) = C(n_G) - C_v(n^v)$;
9 $candidatos = \{(u \in N, v \in N_v), \forall (n_G, u) \in E \wedge (n^v, v) \in E_v\}$;
10 **while** $\exists v \in N_v \wedge v \notin M_N \wedge \#candidatos > 0$ **do**
11 $(u, v) = proximo\_par(candidatos)$;
12 $candidatos = candidatos - \{(u, v)\}$;
13 **if** $C((n_G, u)) \geq C_v((n^v, v)) \wedge caixa(n_G \rightarrow u) \geq caixa(n^v \rightarrow v)$ **then**
14 $C((n_G, u)) = C((n_G, u)) - C_v((n^v, v))$;
15 $M_P((n^v, v)) = M_P((n^v, v)) \cup (n_G, u)$;
16 **if *not*** $VITreeM(G, G_v, u, v, T, T_v, M)$ **then**
17 $C((n_G, u)) = C((n_G, u)) + C_v((n^v, v))$;
18 $M_P((n^v, v)) = M_P((n^v, v)) - (n_G, u)$;
19 **if** $\forall (n^v, v) \in E_v \wedge v \in M_N$ **then**
20 **return** (*verdadeiro*, M);
21 **else**
22 remover todo o mapeamento realizado a partir deste ponto;
23 **return** (*falso, null*);

Inicialmente, o algoritmo define uma tentativa de mapeamento de T_v, enraizada pelo nó n^v, sobre alguma subárvore de T enraizada pelos nós vizinhos de n_G (linhas 1 a 4). Para cada subárvore enraizada pelos vizinhos u de n_G, compara-se os valores da caixa de n^v (capacidade de máquinas virtuais, roteadores e canais de comunicação) aos valores da caixa de u. A função $anterior()$ indica o candidato anteriormente alocado ou o próprio nó quando o algoritmo é iniciado. Esta restrição é um dos pontos chave do algoritmo, pois diminui o espaço de busca do problema, evitando a verificação desnecessária em nós profundos, quando detecta-se que a subárvore de n^v não é comportada pela subárvore de u. Caso a restrição não seja infringida, o algoritmo é executado recursivamente dando continuidade ao mapeamento (linha 3).

Se um mapeamento válido não for encontrado durante o processo descrito, tenta-se o mapeamento do nó n^v sobre o nó n_G (linhas 5 a 8). Se for possível, realiza-se o mapeamento das subárvores enraizadas por n^v sobre as subárvores enraizadas pelos nós vizinhos de n_G (linhas 9 a 18). Para tal, cria-se um conjunto (identificado por $candidatos$) com todos os possíveis pares entre os vizinhos de n_G e os vizinhos de n^v. A ordem em que os pares são selecionados segue um critério pré-estabelecido, como por exemplo *best-fit* que ordena a lista de pares de forma não-decrescente tendo como métrica a capacidade residual resultante da diferença entre as caixas do recurso físico e as caixas do recurso vir-

tual, ou *worst-fit* que ordena de forma não-crescente [Ruck et al. 2014]. Ao final, verifica-se a existência de um mapeamento válido para os vizinhos de n^v; caso contrário, toda alocação realizada a partir da chamada corrente do algoritmo é desfeita. Este algoritmo possui uma complexidade assintótica de $O(n^2log(n))$ para o melhor caso, onde é possível fazer o mapeamento na sua primeira tentativa. Já para o pior caso, sua complexidade é $O(2^nlog(n))$, onde ele irá verificar todas as possibilidades de mapeamento.



(a) Análise do nó v sobre o nó a.

(b) Análise do nó v sobre os vizinhos do nó a.

(c) Mapeamento do nó v sobre o nó a.

(d) Mapeamento do nó w sobre o nó c.

(e) Análise dos nós y e z.

(f) Mapamento dos nós y e z sobre os nós g e f.

Figura 5. Exemplificação do algoritmo VITreeM.

A Figura 5 exemplifica o algoritmo em execução. Inicialmente, a Figura 5(a) descreve a análise do nó $a \in N^T$, da árvore física $T = (N^T, E^T, C^T)$, e do nó $v \in N_v^T$ da

árvore virtual $T_v = (N_v^T, E_v^T, C_v^T)$. Comparando os valores de cada item das duas caixas, observa-se a possibilidade de alocar a árvore T_v sobre a árvore T, pois os valores da caixa do nó a são maiores que os valores da $caixa$ do nó v. O segundo passo (Figura 5(b)), busca um nó vizinho de a capaz de comportar a caixa do nó v (linhas de 1 a 4 do Alg. 1). Observa-se que não existe nenhum nó com recursos suficientes para comportá-lo, pois não há nenhuma caixa de um nó vizinho de a que comporte a caixa de v.

Como não foi possível avançar para os vizinhos de a, VITreeM realiza o mapeamento do nó v sobre o nó a (linhas 7 e 8 do Alg. 1), conforme ilustrado na Figura 5(c). Na sequência, o nó $w \in N_v^T$ é mapeado sobre o nó $c \in N^T$ pois não há vizinhos de c capazes de hospedar o nó w (Figura 5(d)). A mesma situação ocorre com o nó $x \in N_v^T$, não sendo possível avançar sobre nenhum vizinho de $b \in N^T$, pois estes não tem capacidades suficientes para comportá-lo. Devido a esta impossibilidade, x é mapeado sobre o nó b, restando assim apenas o meamento de seus vizinhos $y, z \in N_v^T$ (linhas 9 a 18 do Alg. 1). Como ilustrado pela Figura 5(e), apenas os nós $e, f, g \in N^T$ conseguem comportá-los, não havendo diferença na escolha dos mesmos, pois possuem capacidades iguais. A Figura 5(f) apresenta o estado final do mapeamento.

5. Análise Experimental

O algoritmo foi implementado em *C++*, compilado pelo *g++4.8* com o parâmetro de otimização *-O2* [2]. Os testes foram executados em um equipamento com *AMD Phenom II X4*, 4 GB RAM e sistema operacional Ubuntu 14.04 (linux kernel 3.13).

5.1. Definição das Topologias Físicas e Virtuais

Seguindo o trabalho de [Cavalcanti et al. 2014], as infraestruturas físicas foram baseadas no modelo de três camadas da Cisco [Cisco 2007]. Assim, definiu-se que cada *datacenter* é composto por: *i)* camada de acesso, representada por 4 *racks*, cada um contendo 12 servidores e 1 *switch top-of-rack*, conectados por enlaces de 1 Gbps; *ii)* camada de agregação, com 4 *switches* interconectados por enlaces de 10 Gbps; *iii)* camada de núcleo, com 2 *switches* e enlaces de 10 Gbps. Para a simulação dos testes foram definidos três cenários baseados no *datacenter* descrito, que representam provedores com capacidades distintas. A Tabela 1 resume a quantidade total de nós para cada cenário.

Cenário	# *datacenters*	# servidores	# *switches*	# total
1	3	144	30	174
2	6	288	60	348
3	9	432	90	522

Tabela 1. Composição dos cenários de teste.

As requisições de IVs foram baseadas na topologia *fat-tree*, variando o número de camadas entre 2 e 7. Cada camada contém entre 2 e 5 *switches*, com exceção da última camada que possui entre 5 e 10 MVs. Seguindo trabalhos anteriores [Bays et al. 2012] [Oliveira et al. 2013] [Cavalcanti et al. 2014], cada MV, *switch* ou enlace virtual irá consumir entre 1 e 10% do recurso físico. Para cada cenário físico, foram geradas 100 requisições de IVs distribuídas entre 1 e 150 intervalos discretos (uma

[2] O código da implementação pode ser acessado em *https://bitbucket.org/ramoncs/VITreeM*

IV permanece ativa no máximo por 30 intervalos). Os valores foram distribuídos uniformemente.

5.2. Métricas para Análise

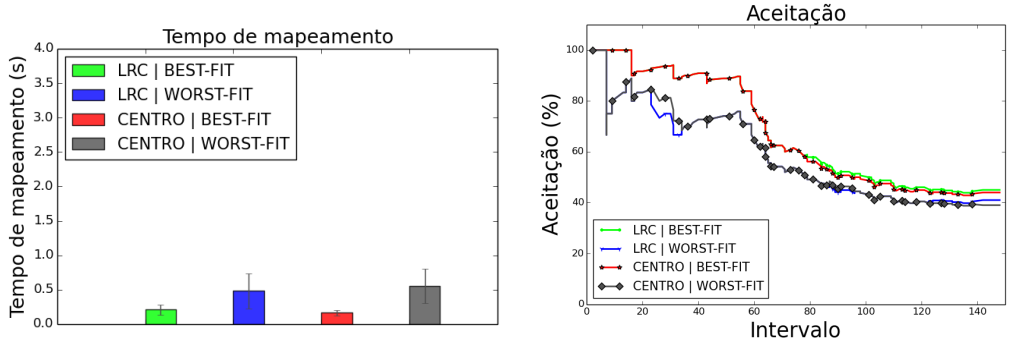
Foram selecionadas quatro métricas para análise de VITreeM: *i)* o *tempo médio de alocação* para uma requisição; *ii)* a *taxa de aceitação*, indicando o percentual de IVs mapeadas com sucesso, definida como a razão entre o número de IVs alocadas e o número de requisições; *iii)* a *fragmentação*, indicando a porcentagem de recursos ativos na infraestrutura física (que estão alocando elementos de alguma IV), definida pela divisão entre o número de recursos ativos e o total de recursos; *iv)* a *carga dos nós e dos canais físicos*, representando o percentual de recursos que estão sendo utilizados, ou seja, a divisão do somatório de capacidades alocadas pelo total existente, permitindo a visualização da capacidade total comprometida na alocação das IVs.

5.3. Análise dos Resultados

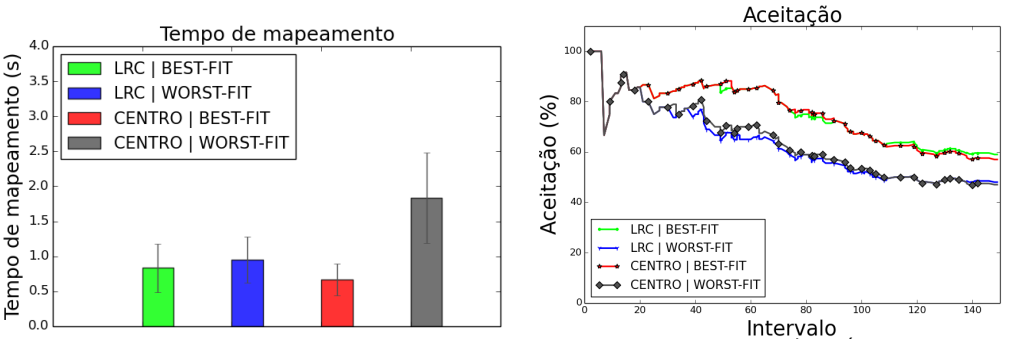
Para realizar uma análise comparativa, os resultados representam o comportamento de VITreeM perante a escolha do tipo de nó de partida (LRC ou centro) e da ordem em que os pares de candidatos são selecionados (*best-fit* ou *worst-fit*). Conforme discutido na Seção 4, a definição destes parâmetros afeta o desempenho dos algoritmos de alocação.

A Figura 6 apresenta os resultados para as métricas de taxa de aceitação (TA) e tempo de execução (TE). As Figuras 6(a), 6(c) e 6(e) resumem o tempo de mapeamento das requisições submetidas aos cenários 1, 2 e 3, respectivamente, bem como o desvio padrão para cada caso. Observa-se que VITreeM obteve um desempenho estável, com tempo médio de alocação abaixo de 3 segundos para todos os cenários, destacando-se as simulações que utilizaram *best-fit* como método de seleção de candidatos, pois apresentam um desvio padrão menor que os demais. O tempo médio de alocação indica a possibilidade de utilização de VITreeM em cenários reais, já que usuários requisitantes aguardariam, em média, poucos segundos. A TA para os cenários 1, 2 e 3 (Figuras 6(b), 6(d) e 6(f)) indica que a utilização de VITreeM com a abordagem de ordenação de candidatos *best-fit* resulta em um maior número de requisições aceitas, independente da abordagem utilizada para definição do nó inicial. Especificamente, na Figura 6(f), as simulações com *best-fit* obtiveram aproximadamente 80% de TA enquanto *worst-fit* obteve aproximadamente 60%. Essa análise acompanha resultados anteriores [Ruck et al. 2014], ressaltando que, para as topologias analisadas, a utilização de LRC [Yu et al. 2008] pode ser substituída pela identificação do nó central sem prejuízos ao provedor hospedeiro.

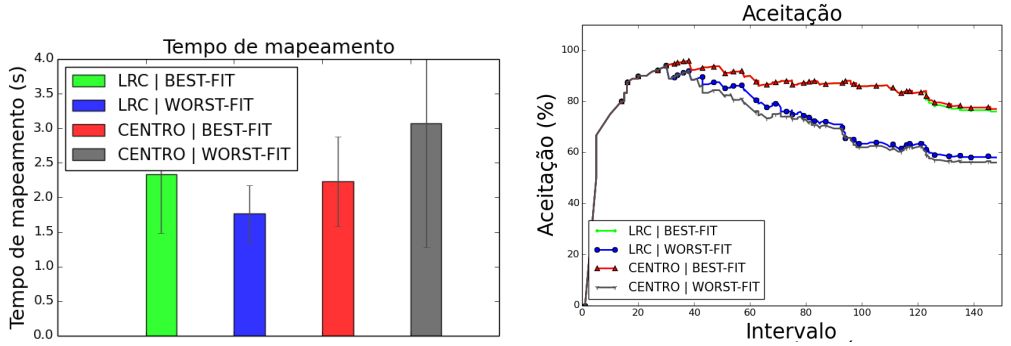
A fragmentação da infraestrutura física (FG) e a carga dos canais de comunicação (CC) são apresentadas na Figura 7. Para FG (Figuras 7(a), 7(b) e 7(c)), observa-se nos cenários 2 e 3 uma maior fragmentação inicial quando o algoritmo *worst-fit* é aplicado. Após a estabilização (próximo dos intervalos 50 e 80 para os cenários 1 e 2, respectivamente), não se observam variações significativas. Analisando o cenário completo, a FG manteve-se estável, aproximadamente 20%, valor considerado baixo quando comparado a trabalhos anteriores [Cavalcanti et al. 2014]. Justifica-se a baixa FG, independente da abordagem de seleção de candidatos, pela natureza do algoritmo, que realiza o mapeamento guiado por uma busca em profundidade na árvore. O espalhamento (a busca em largura), usualmente ocorre apenas próximo das folhas.



(a) Cenário 1 - TE. (b) Cenário 1 - TA.



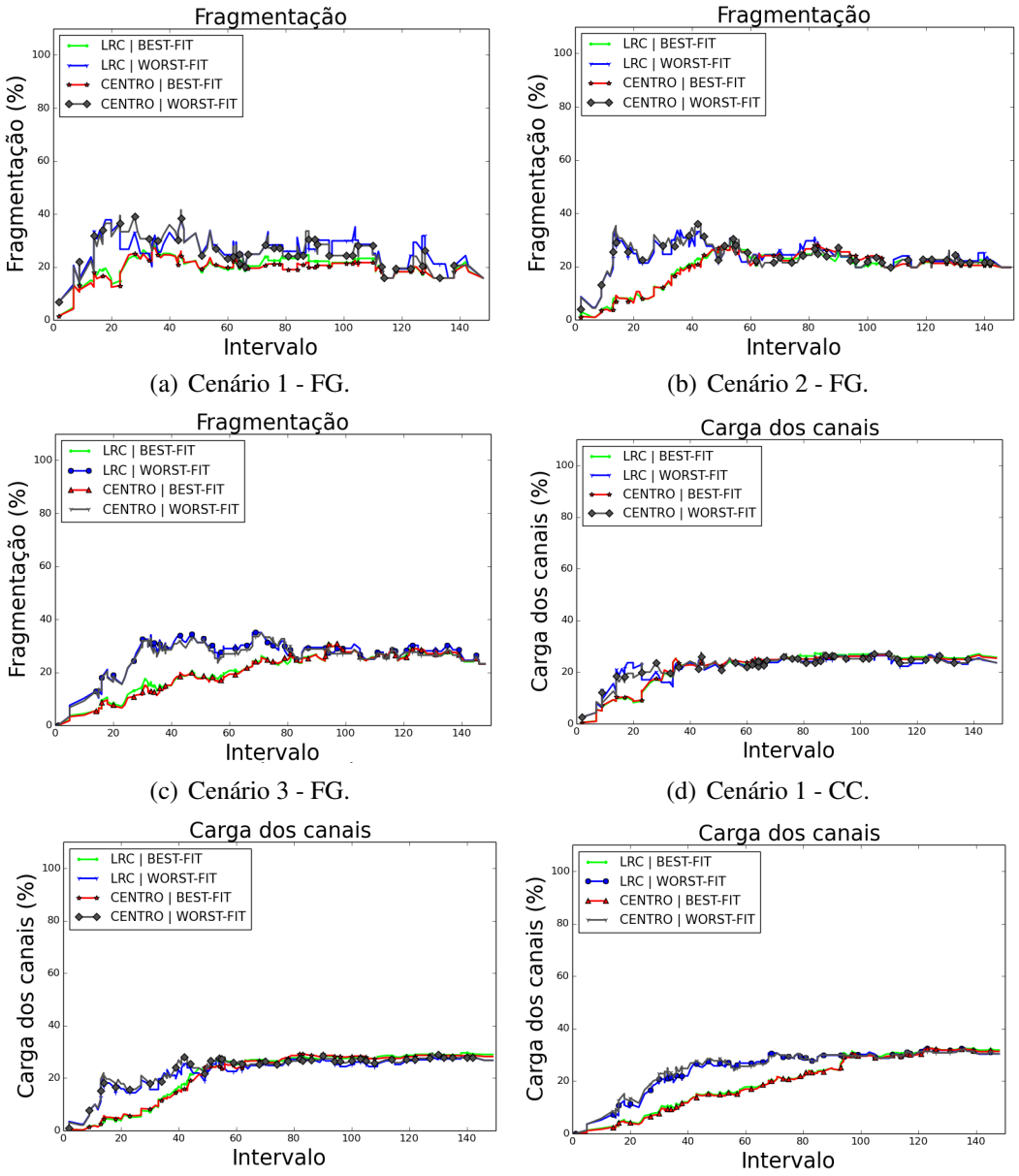
(c) Cenário 2 - TE. (d) Cenário 2 - TA.



(e) Cenário 3 - TE. (f) Cenário 3 - TA.

Figura 6. Resultados das simulações para as métricas de tempo de execução (TE) e taxa de aceitação (TA).

Quanto a carga computacional, as Figura 7(d), 7(e) e 7(f) apresentam a carga dos canais de comunicação (CC). Conforme observado na FG, as simulações que utilizaram *best-fit* para ordenação dos candidatos induziram uma menor CC no início das simulações. Porém, em regime estacionário, os valores estabilizaram em aproximadamente 25%, independente do cenário analisado. Observa-se que, embora VITreeM aproxime os recursos, a topologia física não é sobrecarregada. Por sua vez, os resultados para a métrica da carga dos nós (gráfico omitido) indicam que, em nenhum momento, o algoritmo saturou os recursos computacionais disponíveis. Em suma, os resultados obtidos mostram que a redução do espaço de busca introduzida por VITreeM permite sua aplicação em cenários reais. O algoritmo encontra soluções em tempo viável para diversos cenários, mantendo uma alta TA e induzindo uma baixa FG e CC.



(a) Cenário 1 - FG. (b) Cenário 2 - FG.

(c) Cenário 3 - FG. (d) Cenário 1 - CC.

(e) Cenário 2 - CC. (f) Cenário 3 - CC.

Figura 7. Resultados das simulações para as métricas de fragmentação (FG) e carga dos canais de comunicação (CC).

6. Conclusão

A computação em nuvem é um conceito terminantemente presente nos serviços que são disponibilizados através da Internet, destacando-se a entrega de IVs sob demanda. Dentre as tarefas de gerenciamento, a alocação de recursos físicos para hospedar as requisições de IVs é um problema pertencente a classe NP-Difícil. Nesse contexto, este artigo apresentou VITreeM, um algoritmo heurístico para alocação de IVs baseado em árvores. VITreeM transforma grafos (virtuais e físicos) em árvores e efetua a alocação restringindo o espaço de busca através do agrupamento de informações. A análise experimental indicou uma promissora utilização de VITreeM em diversos cenários, já que este manteve uma baixa

fragmentação do substrato combinada com uma alta taxa de aceitação. Para continuidade do trabalho consideraremos a elasticidade das IVs, o impacto da solução sobre aplicações n-camadas e a possibilidade de alocação de recursos em múltiplos provedores.

Referências

Alkmin, G. P., Batista, D. M., and da Fonseca, N. L. S. (2013). Mapeamento de Redes Virtuais em Substratos de Rede. In *Proc. XXIX - SBRC*.

Baase, S. (2009). *Computer algorithms: introduction to design and analysis*. Pearson Education India.

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 *Proc. ACM/IEEE/IFIP CNSM*.

Cavalcanti, G. A., Obelheiro, R. R., and Koslovski, G. P. (2014). Alocação ótima de recursos para infraestruturas virtuais confiáveis. In *Proc. XXXII - SBRC*.

Cheng, X., Su, S., Zhang, Z., Wang, H., Yang, F., Luo, Y., and Wang, J. (2011). Virtual network embedding through topology-aware node ranking. *SIGCOMM Comput. Commun. Rev.*, 41(2):38–47.

Chowdhury, N. and Boutaba, R. (2010). A survey of network virtualization. *Computer Networks*, 54(5):862–876.

Chowdhury, N., Rahman, M., and Boutaba, R. (2009). Virtual network embedding with coordinated node and link mapping. In *INFOCOM 2009, IEEE*, pages 783–791.

Cisco (2007). Cisco Data Center Infrastructure: 2.5 Design Guide.

Fischer, A., Botero, J., Till Beck, M., de Meer, H., and Hesselbach, X. (2013). Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4).

Gong, L., Wen, Y., Zhu, Z., and Lee, T. (2014). Toward profit-seeking virtual network embedding algorithm via global resource capacity. In *Proc INFOCOM*. IEEE.

Kandula, S., Sengupta, S., Greenberg, A., Patel, P., and Chaiken, R. (2009). The nature of data center traffic: Measurements & analysis. In *Proc. of the 9th ACM SIGCOMM IMC*, New York, NY, USA.

Knuth, D. E. (1968). The art of computer programming: Fundamental algorithms, vol. i.

Lischka, J. and Karl, H. (2009). A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proc. of the 1st ACM Workshop on Virtualized infrastructure systems and architectures*.

Luizelli, M. C., Bays, L. R., Buriol, L. S., Barcellos, M. P., and Gaspary, L. P. (2013). Caracterizando o impacto de topologias no mapeamento de redes virtuais. In *Proc. XXXI - SBRC*.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. In *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology*.

Niranjan Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., and Vahdat, A. (2009). Portland: a scalable fault-tolerant layer 2 data center network fabric. In *ACM SIGCOMM Computer Comm. Review*, volume 39.

Oliveira, R. R., Bays, L. R., Marcon, D. S., Neves, M. C., Buriol, L. S., Gaspary, L. P., and Barcellos, M. P. (2013). DoS-Resilient Virtual Networks through Multipath Embedding and Opportunistic Recovery. In *Proc. ACM SAC*, pages 597–602.

Ricci, R., Alfeld, C., and Lepreau, J. (2003). A solver for the network testbed mapping problem. *SIGCOMM Comput. Commun. Rev.*, 33(2):65–81.

Ruck, D. B., Oliveira, R., and Koslovski, G. P. (2014). Comparação de algoritmos para alocação de Infraestruturas Virtuais. *Revista Brasileira de Computação Aplicada*.

Wang, G., Zhao, Z., Lu, Z., Tong, Y., and Wen, X. (2013). A virtual network embedding algorithm based on mapping tree. In *13th Symp. Comm. and Information Technologies (ISCIT)*, pages 243–247.

West, D. B. et al. (2001). *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

Yu, M., Yi, Y., Rexford, J., and Chiang, M. (2008). Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*.

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.

Armazenamento em Redes Orientadas a Conteúdo baseado em Ranqueamento de Caches

Edson Adriano Maravalho Avelar, Kelvin Lopes Dias.

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
CEP: 50740-540 – Recife – PE – Brasil

{eama,kld}@cin.ufpe.br

Abstract. *In-network caching is one of the main features of the Information Centric Network (ICN). Thus, the adopted caching strategy is crucial for the efficient delivery of content and cache storage. In this paper, we proposed a caching algorithm (RankCache) by exploring the concept of cache ranking. The ranking of each cache is calculated using interval estimation of binomial proportion, where the success parameter is related to requested content popularity. Through simulations, our algorithm is compared with several caching strategies, such as LCE, Fix Probability, Betweenness Centrality, LCD and ProbCache. A performance evaluation was conducted which confirmed the superiority of RankCache in terms of hit rate, hit hop, server hit and overall delay, with gains ranging from 10% to over 30%.*

Resumo. *O armazenamento de dados nos nós da rede é uma das principais características para manter e entregar conteúdos de forma eficiente nas ICN (Information Centric Networks) ou Redes Centradas na Informação. Neste artigo, propomos um algoritmo de armazenamento, chamado RankCache, baseado no conceito de ranqueamento de cache. O ranking de cada cache é calculado usando estimação intervalar da proporção binomial, onde o parâmetro de sucesso está relacionado à popularidade dos conteúdos requisitados. Através de simulações, o algoritmo foi comparado com várias estratégias da literatura, como LCE, Fix, Betweenness Centrality, LCD e ProbCache. A análise de desempenho mostrou a superioridade da proposta em termos de taxa de acerto, número de saltos, acertos no servidor e atraso total, com ganhos que variam de 10% a 30%.*

1. Introdução

Quando a Internet foi idealizada, o principal problema que precisava que ser resolvido era a entrega de dados entre computadores ou entre computadores e terminais. O modelo básico era centrado na comunicação *host-to-host. Esse* modelo reduziu a eficiência dos serviços de distribuição de conteúdo. Como resultado, a rede de comunicação era, e continua a ser, baseada na localização física de hospedeiros [Jacobson 2009, Jacobson 2012].

No entanto, a Internet de hoje está mais preocupado com a conexão entre pessoas, conteúdos e informações. Esses conteúdos incluem *streaming* de áudio e vídeo, redes sociais, jogos online, entre outros. Segundo o documento de previsão da Cisco® [Cisco 2014] para 2013-2018, o tráfego global de vídeo será responsável entre 80% a 90% de todo o tráfego da internet até 2018. Segundo o mesmo relatório, o tráfego gerado por redes de entrega de conteúdo ocupará metade de todo tráfego de vídeo pela

Internet até 2018. Esses números mostram um grande crescimento na busca por conteúdo. Neste novo cenário, a localização física é a que menos importa para os usuários, que querem conteúdo sempre disponível, com maior velocidade possível e melhor qualidade de serviço.

Seguindo essa tendência, as Redes Centradas na Informação [Brito 2012, Torres 2013], tradução adaptada de ICN (*Information Centric Network*), oferecem um novo paradigma onde as operações da rede são focadas no alvo de interesse dos usuários, a informação. Nas ICN, um conteúdo é consumido diretamente através de seu nome e não de sua localização. Diferentemente da abordagem tradicional das redes IP, centrada na identificação e localização de nós, as ICNs usam conceitos mais inovadores como identificação de conteúdo por nome, roteamento baseado em nomes, segurança aplicada diretamente ao conteúdo e armazenamento de dados nos elementos da rede [Ghodsi 2011].

Dentre as várias propostas para ICNs [Ahlgren 2012], a arquitetura CCN (*Content Centric Network*), é uma das que mais se destacou, pois, além da farta documentação, ela possui uma implementação oficial de código fonte aberto. Essa implementação, nomeada CCNx, foi criada e é mantida pela PARC (*Palo Alto Reasearch Center/Xerox*) [CCNx 2014]. Na arquitetura CCN, o nome do conteúdo é desacoplado do endereço físico de seu provedor. O benefício dessa abordagem é a clara separação entre a identificação e a localização do conteúdo. Como nas redes IP não há essa separação, problemas como mobilidade e segurança nunca tiveram soluções amplamente aceitas, devidos os remendos propostos na pilha TCP/IP, como Mobile IP e IPSec.

O CCN utiliza dois tipos de pacotes: *Interest* e *Data*. O consumidor expressa seu interesse inserindo o nome do conteúdo desejado em uma mensagem do tipo *Interest* e a envia para rede. O produtor, ou algum cache no interior da rede, que possui o conteúdo, receberá essa mensagem e enviará de volta ao consumidor uma mensagem do tipo *Data* como resposta. Portanto, essas mensagens possuem uma relação um para um, onde um pacote de interesse satisfaz um de dados se o nome do conteúdo em ambos os pacotes forem equivalentes.

No CCN, os conteúdos são quebrados em pequenos pedaços chamados de *chunks*[1]. Esses pedaços podem ser auto-identificados (*self-indentified*) e auto-autenticados (*self-authenticated*), sem precisar estar associados a um *host* em particular. Além disso, essa arquitetura permite armazenamento dentro dos elementos da rede (*in-network* ou *in-path caching*). Essa característica melhora significativamente a entrega de conteúdos populares (constantemente requisitados), pois os usuários podem receber o conteúdo diretamente de elementos do núcleo da rede, sem que a requisição precise chegar ao provedor do conteúdo. Se um *chunk* do conteúdo requisitado é encontrado no cache de um nó da rede, que não seja o provedor, diz-se que ocorreu um "*acerto*" (*hit*, em inglês) naquele cache. Se um *chunk* não for encontrado em um cache, diz-se que ocorreu um "erro" (*miss*, em inglês) naquele cache.

O armazenamento na rede possui vários benefícios incluindo melhora no desempenho da entrega de conteúdo, diminuição do tráfego na rede, diminuição do atraso dos *chunks* entre outros. No entanto, existem vários desafios que precisam ser

[1] No restante do artigo, para manter o termo consolidado, será usado o termo "*chunk"* para pedaço de conteúdo.

investigados. Um dos mais notáveis é decidir em que caches guardar um *chunk*, de modo a maximizar a probabilidade de ocorrer um *acerto*, diminuir a redundância de *chunks* nos caches e reduzir o atraso de entrega dos conteúdos.

Nesse contexto, este artigo propõe um esquema de armazenamento de *chunks* chamado *RankCache*. O objetivo dessa proposta é prover entrega eficiente de conteúdo e armazenamento otimizado com redução de sobrecarga de gerenciamento e cooperação implícita entre os nós da rede. O algoritmo do RankCache distribui os *chunks* considerando o *ranking* de cada cache, que no CCN pode ser chamado de *Content Store* (CS), *Content Router* (CR) ou apenas cache. As principais características do *RankCache* são:

a) **Ranking baseado na Popularidade**: O algoritmo ajusta o valor dos ranking de cada cache em um dado caminho. O cache com maior valor de ranking possuirá a maior probabilidade de armazenar um *chunk* do que um cache com menor ranking. Esse valor é calculado usando o WSI (*Wilson Score Interval*) [Wilson 1927], que é uma fórmula para calcular o intervalo de confiança para proporção binomial (sucesso versus fracasso). Na proposta, a variável de sucesso é baseada no número de *acertos* de um cache. Quanto mais *chunks* populares passar pelo cache, maior o número de *acertos* dentro dele, e consequentemente, maior sua probabilidade de armazenar outros *chunks*.

a) **Cache Decentralizado**: No RankCache, a decisão de guardar ou não um *chunk* é feita de forma independente. Cada cache calcula sua probabilidade de armazenamento sem a interferência de outros caches. Assim, não há nenhum gerenciamento centralizado. Dessa forma, a proposta é escalável, pois não requer conhecimento a priori da topologia da rede.

O restante do artigo é organizado da seguinte forma. A Seção 2 apresenta os trabalhos que serão utilizados como comparação com o RankCache. A Seção 3 detalha o funcionamento do esquema proposto. Na Seção 4, a avaliação de desempenho, via simulação, mostra a superioridade da proposta e a Seção 5 conclui o artigo.

2. Trabalhos Relacionados: Políticas de Decisão de Cache

As estratégias de armazenamento na rede podem ser divididas em duas políticas: de substituição (*replacement*) e de decisão (*decision*). A primeira define qual *chunk* armazenado será removido quando o cache estiver cheio. A segunda está relacionada à decisão de guardar ou não um *chunk* que atravessa o nó. Existem várias políticas de substituição de *chunks*, porém a mais utilizada em trabalho de cache é a LRU (*Least Recently Used*)[Chai 2013, Psaras 2011], que remove o item menos usado do cache. Essa política também foi recomendada pelos autores do CCN [Jacobson 2009].

Existem alguns esforços para propor soluções no contexto de políticas de decisão de cache. A seguir, serão detalhadas as propostas mais citadas e que farão parte da avaliação comparativa deste trabalho.

A. LCE (*Leave Copy Everywhere*)

Esta é a proposta mais usada em caches multi-níveis. Após detectar uma cópia do *chunk* requisitado, uma cópia é guardada em todos os caches intermediários no caminho de volta para o usuário requisitante. Apesar de simples, esse esquema

apresenta grandes desvantagens, como o alto nível de redundância de *chunks* na rede. Porém, é bastante usada como base para comparação entre propostas [Laoutaris 2004, Chai 2013].

FixProb (ou apenas Fix) é uma versão randômica do LCE. Cada nó intermediário ao longo do caminho entre o consumidor e o provedor é um candidato possível para guardar uma cópia do objeto requisitado. Os caches intermediários podem guardar os objetos com uma probabilidade *p*. Nota-se que o FixProb com *p* igual a 1 (100%) é idêntico ao LCE. Para fins de comparação usou-se *p* igual a 0.3 e 0.7, mesmos valores usados em alguns trabalhos [Psaras 2012].

B. LCD (*Leave Copy Down*)

É um algoritmo, proposto por [Laoutaris 2004], onde uma cópia do conteúdo requisitado é armazenada no cache após um *acerto*. Sempre que é encontrado um objeto dentro do cache, ele é armazenado apenas um cache após o *acerto*. O algoritmo verifica o número de saltos *H* que o *chunk* percorreu; apenas se $H = 1$ o cache armazena o *chunk*. O LCD é mais "conservador" que o LCE, pois ele precisa de várias requisições para levar o conteúdo para perto do consumidor. Por outro lado, esse algoritmo reduz o número de cópias redundantes de *chunks* na rede.

C. Betweenness Centrality

Uma importante classe de políticas de decisão são as baseadas no conceito de centralidade dos nós [Izquierdo 2006]. O princípio fundamental desta estratégia é que, se um nó se encontra ao longo de um grande número de caminhos de entrega de conteúdo, é mais provável que consiga um *acerto*. Uma vez que existem várias estratégias de centralidade, a nossa comparação considerará o algoritmo *Betweenness Centrality* [Chai 2011, Chai 2013], por ser a que apresenta melhor desempenho e maior facilidade de implementação.

D. ProbCache

Apresentado por [Psaras 2012], o ProbCache é uma algoritmo que explora o conceito de capacidade de armazenamento do caminho (*path-caching capacity*). Este valor é calculado através da medição da capacidade de armazenamento médio dos caches no trajeto do pacote. A proposta introduz dois novos cabeçalhos nas mensagens CCN, o TSI (*Time Since Inception*) e o TSB (*Time Since Birth*). Cada roteador aumenta o valor TSI dos pacotes de solicitação em um. O provedor do conteúdo atribui o valor TSI que vê na mensagem de requisição para a mensagem de resposta. Cada roteador aumenta o valor TSB da mensagem de resposta também em um. Assim, quando o conteúdo viaja de volta para o cliente, o TSI tem um valor fixo e indica o comprimento do caminho desse fluxo específico, enquanto o valor de TSB indica o número de saltos que o conteúdo da mensagem de respostas percorreu desde o *acerto*.

3. Algoritmo de Armazenamento em redes CCN baseado no Ranking do Cache.

As melhores propostas de políticas para a tomada de decisão levam em conta a popularidade de pedaços de conteúdo (*chunk*). Um *acerto* no cache envolve nada mais do que achar conteúdos requisitados anteriormente por outros usuários e que estejam armazenados. A probabilidade de um sucesso é diretamente proporcional à popularidade

de um *chunk*. Uma política de decisão de cache eficiente deve ser capaz de prever quando essas requisições repetidas ocorrerão. No RankCache, proposto neste trabalho, o armazenamento dos *chunks* é proporcional ao ranking de cada cache. O ranking de um dado cache é calculado com base no número de requisições de *chunks* repetidos que passam através do cache.

O problema de calcular o ranking de vários caches independentes, usando a proporção de número de *acertos* versus número de *erros*, não é simples como usar um sistema de classificação médio (número de *acertos* dividido pelo número total de pacotes). Por exemplo, com classificação média, um cache com $3x10^3$ *acertos* e $6x10^3$ pacotes no total (*acertos* + *erros*) terá o mesmo valor de ranking que um cache com 6 *acertos* e 12 *chunks*; e isso não é uma representação justa. Neste trabalho, usou-se um cálculo mais sofisticado para criar o ranking de caches. O cálculo leva em consideração dois fatores: a proporção binomial e o valor de proximidade.

Existem algumas fórmulas para cálculo de proporção binomial com Intervalo de Confiança [Edwardes 1994]. Neste artigo, utilizou-se a fórmula chamada *Wilson Score Interval*, proposta por Edwin B. Wilson [Wilson 1927], por sua simplicidade e velocidade de processamento. A fórmula de Wilson foi adaptada neste artigo para calcular o ranking do cache usando o número de *acertos* em um dado cache. Essa fórmula, denominada Proporção Binomial do *i*-ésimo cache (PBc_i), é dada por:

$$PBc_i = \frac{\rho i + \frac{1}{2ni} z^2_{1-\alpha/2} \pm z_{1-\alpha/2} \sqrt{\frac{\rho i(1-\rho i)}{ni} + \frac{z^2_{1-\alpha/2}}{4ni^2}}}{1 + \frac{1}{ni} z^2_{1-\alpha/2}}$$

Onde:

- $\boldsymbol{\rho i}$ é o número de *acertos* observados no cache *i*.
- $\boldsymbol{ni}$ é o número total de *chunks* que passam pelo cache *i*.
- $\boldsymbol{z^2_{1-\alpha/2}}$ é o valor padronizado da distribuição normal para um dado Intervalo de Confiança.

O cálculo do ranking considera a contagem de ocorrências de sucesso como uma amostra estatística a ser estimada. O algoritmo infere, em um intervalo de confiança, um valor de cache baseado na estimativa da ocorrência de *acertos* naquele cache. Usou-se como Intervalo de Confiança o valor 0,95 (95%), o que dá um valor padronizado ($\boldsymbol{z^2_{1-\alpha/2}}$) igual a 1,96.

Além da inferência de proporção binomial, elaboramos neste artigo um cálculo do ranking usando o que denominamos de Fator de Proximidade (*FP*). Esse fator aumenta a probabilidade de um *chunk* ser armazenado perto do consumidor. Estudos mostraram que os conteúdos tendem a ser armazenados por mais tempo nas bordas da rede [Psaras 2011]. O FP ajuda a fórmula do ranking a modelar esse comportamento. O Fator de Proximidade do i-ésimo cache (FPc_i) é calculado por:

$$FPc_i = \frac{\sum_{j=1}^{Hc} H_j}{\sum_{i=1}^{Ht} H_i}$$

Onde:

- $\boldsymbol{H_c}$ é o número atual de saltos após um *acerto*
- $\boldsymbol{H_t}$ é o número de saltos totais do pacote até a ocorrência do *acerto*.
- $\boldsymbol{H_j}$ e $\boldsymbol{H_i}$ são valores somados a cada salto dos pacotes. Alguns saltos podem ter pesos por algum razão, porém, neste trabalho, foram usados saltos unitários.

Para exemplificar o funcionamento do FPc_i, supõe-se que um *chunk* é encontrado no sétimo cache de um dado caminho. No primeiro salto do pacote de dados, FPc_7 = 1/7, no segundo salto, FPc_6 =2/7, no terceiro, FPc_5 = 3/7 e assim por diante, até chegar ao nó requisitante, que não entra no cálculo, ou seja, nunca FPc_i será igual a 7/7. Percebe-se que o FPc_i é incrementado com a proximidade do nó requisitante, isso ajuda a aumentar a probabilidade de armazenamento na borda da rede.

O valor de H_c pode ser obtido através do campo TTL (*Time To Live*) da mensagem CCN. Para possibilitar todos os cálculos necessários, o RankCache requer a adição apenas do H_t no cabeçalho da mensagem CCN. Depois de um *acerto*, o valor de H_t pode ser calculado através do valor de H_c e será gravado na mensagem de resposta.

A fórmula final para o cálculo da probabilidade de guardar o *chunk* no i-ésimo cache é dada pela seguinte equação:

$$PRc_i = PBc_i * FPc_i$$

O Algoritmo 1 descreve como a função de decisão utiliza a fórmula PRc_i para o armazenamento de um *chunk*. Essa função recebe como parâmetro a mensagem CCN de chegada e retorna sua decisão como um valor booleano.

Algoritmo 1: Função de Decisão do RankCache

```
1.  Entrada: Mensagem CCN //recebe como parâmetro a mensagem CCN :
2.  Saída: booleano decisao // Retorna um valor lógico indicando a decisão.

3.  decisao ← falso.
4.
5.  //Define acertos como o número total de acertos deste cache. Ele precisa ser um
    contador com escopo global.
6.  acertos ← Acertos Totais
7.
8.  n ← n + 1 //Incrementa o número total de chunks n.
9.  obtém o valor de Hc da mensagem CCN;
10. obtém o valor de Ht da mensagem CCN;
11.
12. Rp ← Hc/Ht
13. //PBc também é global para este cache, só é atualizado em um intervalo de tempo
    pré-definido para evitar overhead de atualização.(Linhas 27-30)
14. PRc ← PBc * Rp
15.
16. //Gera um número randômico x
17. x ← Número Randômico entre 0 e 1.
18. se x é menor que Prc então:
19.     decisao ← verdadeiro;
20. fim se
21.
```

```
22. se decisao é verdadeiro então:
23.     Incrementa o valor do número de acertos
24. fim se
25.
26. //atualiza o rank do cache.
27. se o intervalo de atualização do Rc foi alcançado então:
28.     phat ← 1.0*acertos/n;
29.     PBc ← (phat + z*z/(2*n) - z * sqrt((phat*(1-phat)+z*z/(4*n))/n))/(1+z*z/n);
30. fim se
31.
32. retorna decisao
```

Primeiramente, são obtidos os valores H_c e H_t através da mensagem CCN (linhas 9-10). Posteriormente, é calculada a probabilidade de armazenamento do *chunk*. Nota-se que, no algoritmo 1, "PB_c" é uma variável global e é inicializada com 1 fora da função. Se a decisão de guardar o *chunk* for positiva, a variável "*acertos*", que guarda o número de *acertos*, é incrementada (linha 23). O ranking do cache é atualizado, através do calculo do *"PBc"*, em intervalos previamente definidos, para evitar o overhead de atualizações desnecessárias (linha 29). Ao final das operações, a função retorna a decisão de armazenamento.

Esse procedimento, detalhado pelo algoritmo 1, é invocado por uma entidade que gerencia as políticas de cache. Caso a decisão de armazenamento seja positiva e não haja mais espaço no cache, o algoritmo de reposição entra em ação para trocar um *chunk* armazenado pelo que acabou de ser escolhido. A Seção 4 apresenta a avaliação do algoritmo proposto.

4. Avaliação

Para avaliar o RankCache, foi desenvolvido um simulador de rede CCN. Ele foi implementado no framework OMNET++[Varga 2001] e foi baseado em algumas ideias de dois outros simuladores; o ccnSim [Chiocchetti 2013] e CCN-Lite [CCN-lite 2013], ambos implementados no OMNET++. O primeiro é um simulador em nível de *chunk* e possui um alto grau de abstração, por isso, torna-se difícil a avaliação de políticas de roteamento e de distribuição de prefixos; e métricas próximas da camada de rede como atrasos, perdas de pacotes e vazão.

Por outro lado, o CCN-Lite não possui a implementação da camada de núcleo CCN, onde residem a PIT (*Pendent Interest Table*), a FIB (*Forwarding Information Base*) e o CS (*Content Store*). No CCN-Lite, o núcleo CCN é implementado fora do OMNET++, de modo que possa ser aproveitado por outras plataformas como Linux e Raspberry Pi. Apesar de ser uma vantagem em alguns casos, isso dificulta bastante a implementação de novas propostas que envolvam o núcleo CCN, como roteamento, políticas de decisão e de armazenamento, por exemplo. No nosso simulador, foi implemento todo o núcleo CCN, incluindo as tabelas PIT, FIB e CS. Além disso, ele implementa o protocolo OSPFN [Wang 2012] para roteamento e distribuição de prefixos. Além do mais, permite extração de métricas de rede como atraso, perdas de pacote e vazão.

O experimento foi conduzido de acordo com a Tabela 1. O LRU foi usado como política de substituição. A distribuição Zipf foi empregada como modelo de

popularidade de conteúdo. O Zipf é uma das distribuições mais usadas para modelar esse comportamento e é, de acordo com alguns estudos, a que melhor explica as estatísticas de download de conteúdo de serviços de vídeo sob demanda, como o Youtube [Cha 2009].

Tabela 1. Configuração dos parâmetros dos experimentos.

Networks	27 nodes (Rede IPÊ)[RNP 2014]
Atrasos na rede.	10-100ms (uniforme)
Intervalo de Atualização (PB*c*)	15s
Packet Capacity	50 pacotes
Transmission rate	exponencial, média = 1/80
Tamanho do catálogo[3]	10^5 files
Tamanho médio dos conteúdos	100 *chunks*
Tempo de simulação	10^3s

Foi utilizada, como cenário de simulação, a topologia da rede IPÊ [RNP 2014]. Ela possui 27 nós distribuídos, através de PoPs (*Points of Presence*/Pontos de Presença), por todos os estados brasileiros. A Figura 1 mostra a topologia implementada no simulador e usada nas simulações.

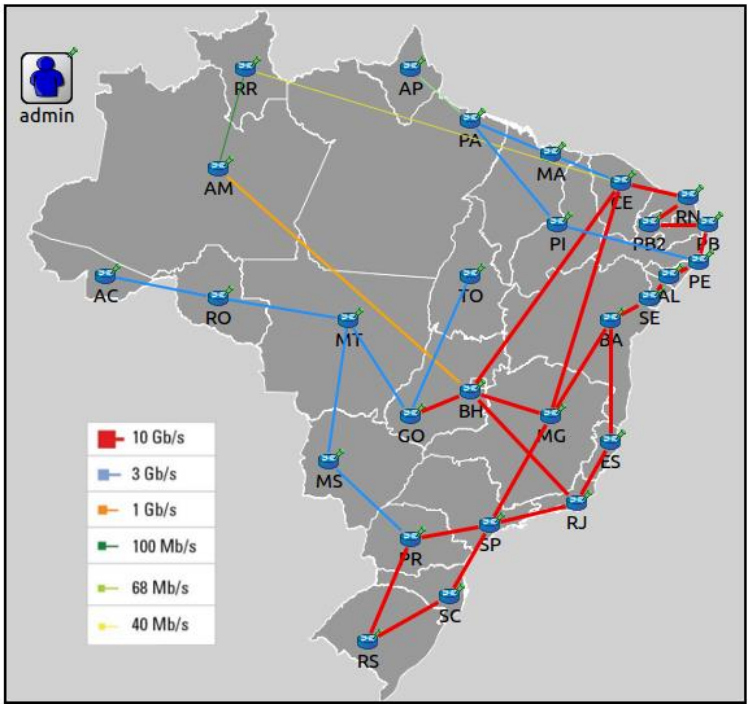


Figura 1. Topologia da Rede IPÊ implementada no simulador

Para todos os experimentos foram feitas 30 simulações. Os valores apresentados nas Figuras 2, 3, 4 e 5, foram obtidos com o expoente da distribuição Zipf iguais a 0.5.

[3] Faz referência ao número de conteúdos únicos.

Quanto menor o Zipf, menor o número de conteúdos populares no catálogo [Psaras 2012].

A Figura 2 apresenta a taxa média de *acertos* em relação à variação da capacidade dos caches. Como pode ser visto na figura, o RankCache possui uma taxa de *acertos* maior que todas as outras propostas. É superior[4] em aproximadamente 15% em relação à média do Fix0.3 e mais de 17% em relação à média do ProbCache.

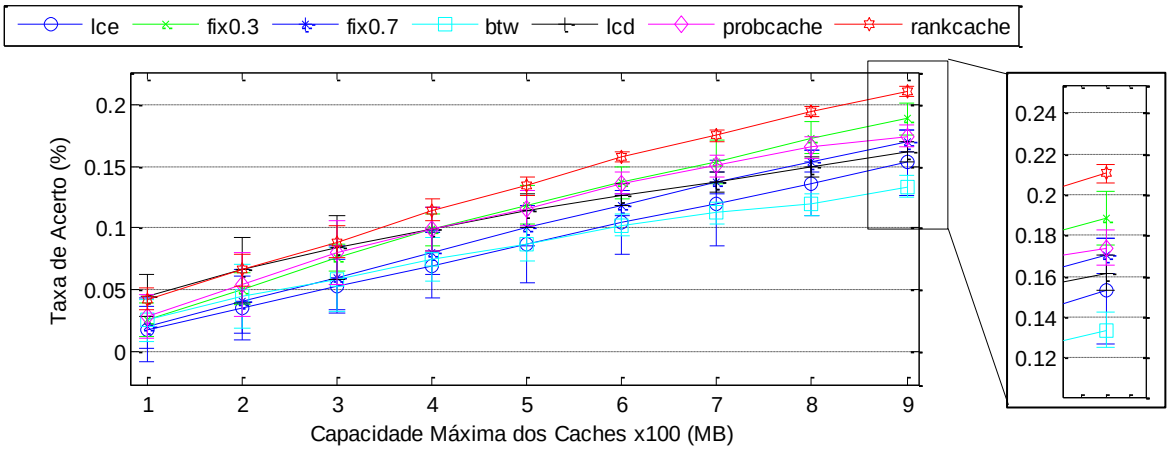


Figura 2. Taxa de *Acertos* comparada com a capacidade dos caches variando de 100 MBytes a 900 MBytes.

A Figura 3 mostra o número de saltos médios necessários para ocorrer um *acerto*. Essa métrica está relaciona à distância média entre o usuário que requisita o conteúdo e onde o conteúdo é encontrado. Quanto menor é o valor dessa métrica, menor é o caminho percorrido pela mensagem de requisição e de resposta. Além disso, mais rápido o usuário consegue o conteúdo requisitado. O número de saltos médio do RankCache, com 900Mb de cache, foi de aproximadamente 2,8 saltos, contra 3,2 saltos da segunda melhor proposta (Probcache). Nessa Figura, o LCE possui o pior desempenho, isso porque o LCE armazena todos os pacotes de forma indiscriminada e sem distinção de nível de popularidade, proximidade ou outro tipo de característica.

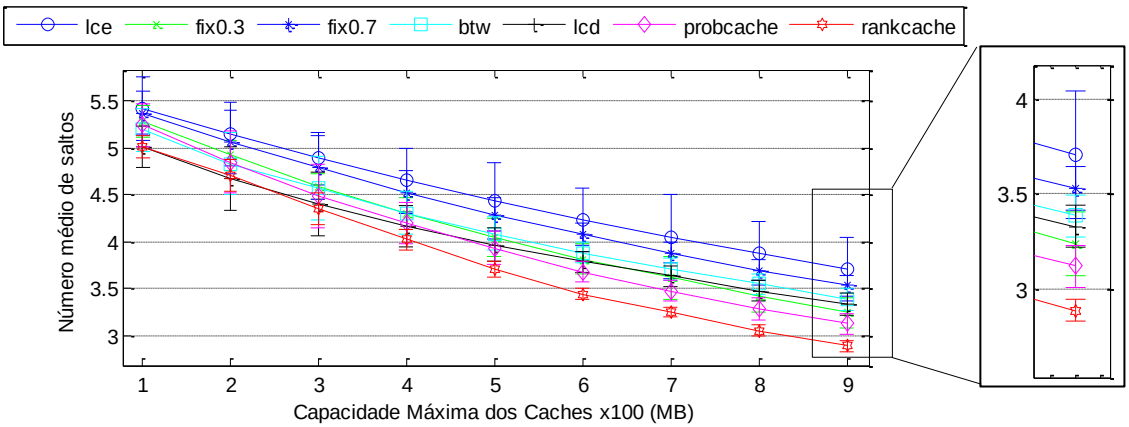


Figura 3. Número de saltos até um *Acertos* em relação à capacidade dos caches variando de 100 MBytes a 900 MBytes.

[4] No artigo, toda a comparação entre propostas é feita em relação à média dos dados.

Outra importante métrica para avaliar o desempenho de algoritmos de cache é a quantidade de *acertos* no servidor, ou seja, no provedor do conteúdo. Quando maior o número de *acertos* no servidor menor é a eficiente do algoritmo em distribuir os *chunks* pela rede. A redução do tráfego total da rede é um dos benefícios do baixo número de *acertos* no servidor, uma vez que, menos pacotes precisaram percorrer todo o caminho entre o consumidor e o provedor. Como apresentado na Figura 3, o RankCache também se mostrou superior nesse experimento. Ele foi o algoritmo que menos atingiu o servidor nas simulações, menor até que o ProbCache, o qual foi desenhado para otimizar essa métrica [Psaras 2012]. Com 900Mbytes de cache, o RankCache foi 17% superior em relação ao ProbCache, segundo mais eficiente.

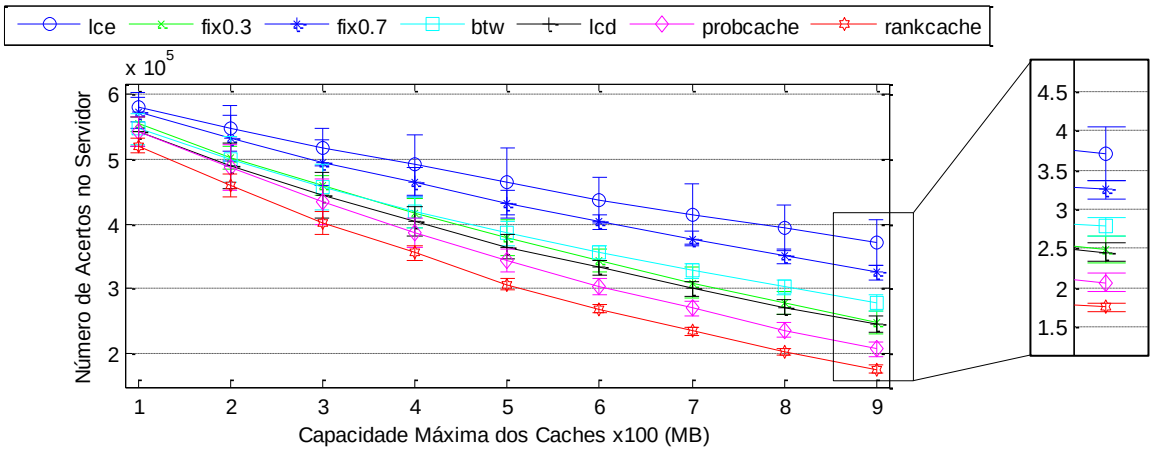


Figura 4. Número de *Acertos* no servidor em relação à capacidade dos caches variando de 100 MBytes a 900 MBytes.

O atraso dos pacotes é outra métrica importante que reflete diretamente no QoS (*Quality of Service*) e o QoE (*Quality of Experience*) das aplicações dos usuários finais. Um baixo atraso é requisito mínimo de serviços de tempo real como streaming de áudio e vídeo. A avaliação mostrou que o RankCache possui o menor atraso médio total. A Figura 5 mostra o tempo de ida e volta (RTT-*Round Trip Time*) das mensagem CCN na rede. Como o RankCache provê uma alta taxa de *acerto* e um baixo número de *acertos* no servidor, o conteúdo é entregue o mais rápido possível, o que reduz o atraso total dos pacotes. O ganho, neste caso, é de 10% em relação ao segundo melhor, ProbCache.

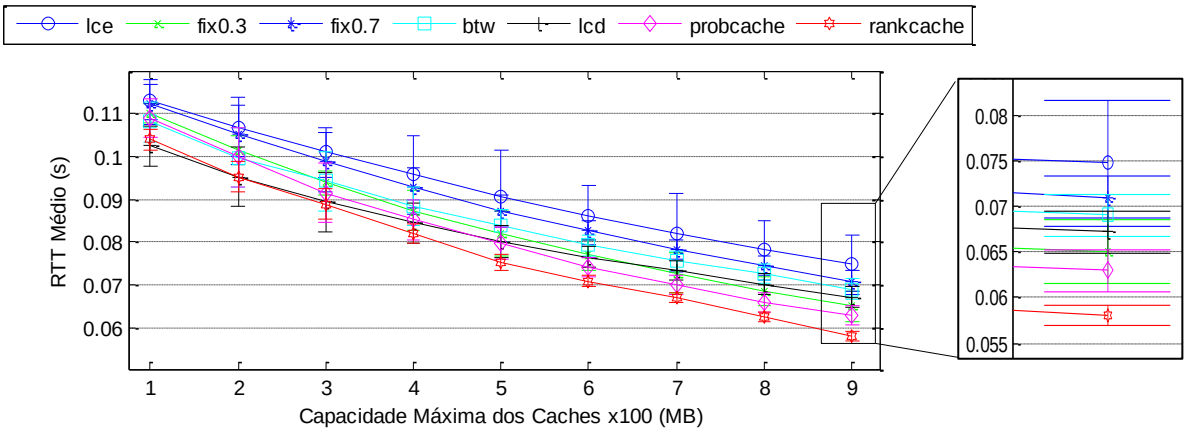


Figura 5. RTT médio em relação à capacidade dos caches variando de 100 MBytes a 900 MBytes.

Na distribuição Zipf, um alto valor do expoente alfa (α) representa um tamanho pequeno de conteúdos com uma popularidade similar. Por outro lado, um valor baixo de α representa um grande número de conteúdos com a mesma popularidade. A Figura 6 mostra o número de *acertos* no servidor em relação à variação do expoente Zipf entre 0.5 e 1.0. O RankCache tem vantagem quando α é pequeno, uma vez que suas características levam em conta a popularidade dos *chunks*. São nos piores casos, onde o número de chunks populares é menor, que o algoritmo se destaca mais. O RankCache tem o melhor desempenho, em torno de 17% com α = 0.5 e 9% com α = 1.0, em relação ao segundo melhor, Probcache.

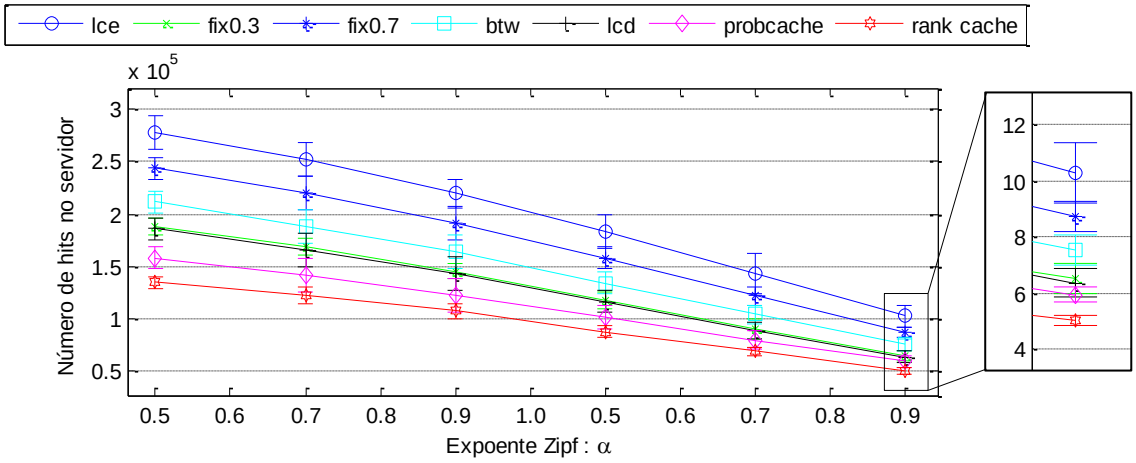


Figura 6. Número de *acertos* no servidor em relação ao expoente Zipf α, onde α varia de $5x10^{-1}$ a $10x10^{-1}$. Tamanho do Cache igual a 900MB

Os resultados que mostram o RTT médio de todos os pacotes em relação ao expoente do Zipf são apresentados na Figura 7. O ganho do RankCache foi de 10% comparado com o Fix03, segundo menor RTT médio.

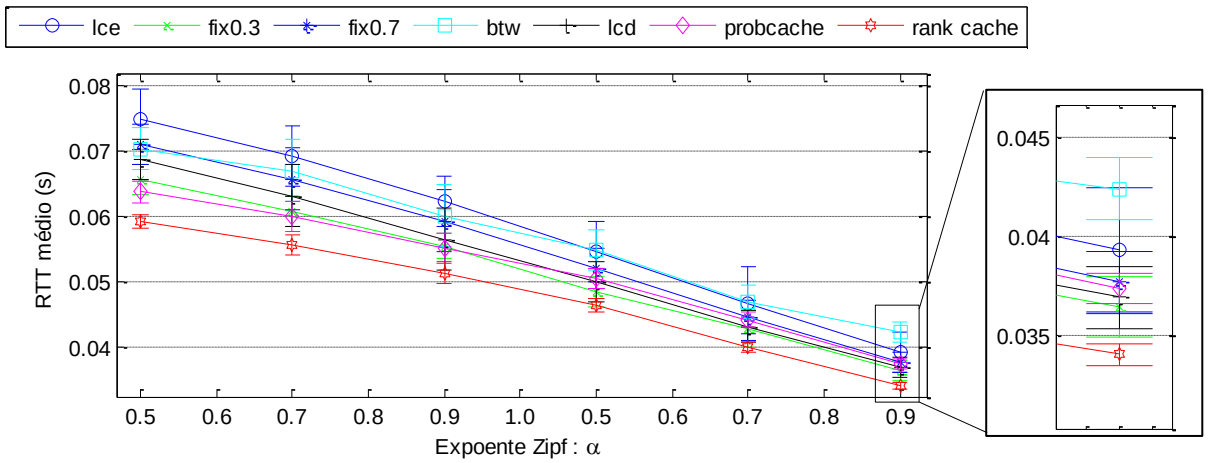


Figura 7. RTT médio em relação ao expoente Zipf α, onde α varia de $5x10^{-1}$ a $10x10^{-1}$. Tamanho do Cache igual a 900MB

Para verificar a natureza distribuída dos algoritmos testados, foi introduzido um modelo de erro de bits igual a $1x10^{-6,}$ uniformemente distribuído. Para essa simulação

foram feitos 15 repetições. A Figura 8 mostra os resultados do número médio de pacotes perdidos na rede. Com um modelo de erro aplicado na rede, perdas de pacotes são inevitáveis. Quanto maior o caminho percorrido, maior a probabilidade de o conteúdo ser perdido por erros na transmissão. O RankCache apresenta menor número de pacotes perdidos, isso se deve ao fato dele distribuir os *chunks* de forma mais eficiente na rede, de modo que as requisições não precisam percorrer longos caminhos para obter o conteúdo. Neste caso, o ganho do RankCache foi de 33% se comparado com o LCD, segundo melhor.

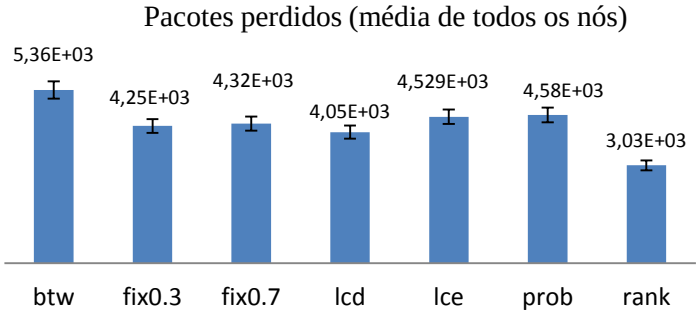


Figura 8. Média dos pacotes perdidos em 15 simulações com modelo de erro de bit igual a $1x10^{-6}$

5. Considerações finais

Neste artigo, foi apresentado um novo esquema de armazenamento em nós da rede chamado RankCache. A proposta cria um ranking de caches no caminho do fluxo de pacotes. Os caches com maior ranking possuem a maior probabilidade de armazenar um determinado pacote. Os rankings são calculados através do intervalo de confiança da proporção binomial, usando a equação de Wilson, onde o valor de interesse é o número de vezes que um conteúdo foi encontrado nos caches intermediários.

A proposta foi avaliada e comparada com várias outras da literatura como *Leave Copy Everywheare*, Fix, *Betweenness Centrality*, *Leave Copy Down* e ProbCache. Em todos os casos, o RankCache se mostrou superior nas métricas avaliadas, com ganhos que variam de 10% a 30%, em relação ao segundo melhor de cada métrica.

Redes Orientadas a Conteúdo é uma área ativa de pesquisa e ainda existem várias lacunas que precisam ser preenchidas como mobilidade, esquemas de nomeação, roteamento, armazenamento entre outras [Kurose 2014].

O artigo deixou algumas questões em aberto, como o desempenho do algoritmo em vários outros cenários; uso de caches heterogêneos, ou seja, com diferença expressiva de tamanho; estudo sobre o intervalo de atualização do PBc_i e a avaliação do custo computacional do cálculo do rank. Todas essas questões serão abordadas em trabalhos futuros.

O próximo passo do trabalho é aprofundar a análise do RankCache implementando-o em redes reais ou emuladas [Cabral 2013]. Será investigado também, o impacto do RankCache em redes móveis através de avaliações com métricas de QoE (*Qualidade de Experiência*).

Agradecimentos

Este trabalho foi parcialmente financiado pela Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE), através do processo IBPG-0829-1.03/12.

Referências

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., & Ohlman, B. (2012). A survey of information-centric networking. Communications Magazine, IEEE, 50(7), 26-36.

Brito, G. M., Velloso, P. B., & 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.

Cabral, C., Rothenberg, C., Magalhães, M.: (2013) "Mini-CCNx: prototipagem rápida para Redes Orientadas a Conteúdo baseadas em CCN", In Salão de Ferramentas do SBRC 2013.

Chai, W. K., Wang, N., Psaras, I., Pavlou, G., Wang, C., de Blas, G. G., ... & Hadjioannou, E. (2011). Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services. Communications Magazine, IEEE, 49(3), 112-120.

Chai, W. K., He, D., Psaras, I., & Pavlou, G. (2013). Cache "less for more" in information-centric networks (extended version). Computer Communications, 36(7), 758-770.

Cha, M., Kwak, H., Rodriguez, P., Ahn, Y. Y., & Moon, S. (2009). Analyzing the video popularity characteristics of large-scale user generated content systems. IEEE/ACM Transactions on Networking (TON), 17(5), 1357-1370.

Chiocchetti, Raffaele, Rossi, Dario and Rossini, Giuseppe, ccnSim: an Highly Scalable CCN Simulator (2013). In IEEE International Conference on Communications (ICC).

CCN-lite, "Lightweight CCN Simulator" (2013). Disponível em: http://www.ccn-lite.net/. Acessado em: nov/2014

CCNx (2014). "A CCN Implementation by Palo Alto Research Center (PARC)". Disponível em: http://www.ccnx.org/. Acessado em: nov/2014.

Cisco Visual Networking Index: "Forecast and Methodology, 2013–2018".(2014) Disponível em: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf. Acessado em: nov/2014

Edwardes, M., & Vollset, S. E. (1994). Confidence intervals for a binomial proportion. Statistics in medicine, 13(16), 1693-1698.

Ghodsi, A., Koponen, T., Rajahalme, J., Sarolahti, P. e Shenker, S. (2011). Naming in content-oriented architectures. Em ACM SIGCOMM Workshop on Information-Centric Networking - ICN, páginas 1–6.

Izquierdo, L. R., & Hanneman, R. A. (2006). Introduction to the formal analysis of social networks using mathematica. University of California, Riverside.

RNP (2014). “Rede Nacional de Pesquisa/Arquitetura da Rede IPÊ”. Disónível em: http://www.rnp.br/servicos/conectividade/rede-ipe. Acessado em: nov/2014

Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N. e Braynard, R. (2009). “Networking named content”. International Conference on emerging Networking EXperiments and Technologies - CoNEXT.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M., Briggs, N. e Braynard, R. (2012). “Networking Named Content”. Communications of the ACM, 55(1):117–124.

Kurose, Jim. (2014) "Information-centric networking: The evolution from circuits to packets to content." Computer Networks. p. 112-120.

Laoutaris, N., Syntila, S., & Stavrakakis, I. (2004). Meta algorithms for hierarchical web caches. In Performance, Computing, and Communications, 2004 IEEE International Conference on (pp. 445-452). IEEE.

Psaras, I., Clegg, R. G., Landa, R., Chai, W. K., & Pavlou, G. (2011). Modelling and evaluation of CCN-caching trees. In NETWORKING 2011 (pp. 78-91). Springer Berlin Heidelberg.

Psaras, I., Chai, W. K., & 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 (pp. 55-60). ACM.

Torres, J. V., Ferraz, L. H. G., Duarte, O. C. M. B. (2013). Redes orientadas a conteúdo baseadas em controladores hierárquicos. XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC.

Varga, A. (2001). The OMNeT++ discrete event simulation system. In Proceedings of the European Simulation Multiconference (ESM’2001) (Vol. 9, p. 185). sn.

Wang, L., Hoque, A. K. M. M., Yi, C., Alyyan, A., & Zhang, B. (2012). OSPFN: An OSPF based routing protocol for Named Data Networking. University of Memphis and University of Arizona, Tech. Rep.

Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association, 22(158), 209-212.

Avaliação de Balanceamento de Carga Web em Redes Definidas por Software

Cristiane P. Rodrigues[1], Leonardo C. Costa[1]
Marcos Augusto M. Vieira[2], Luiz Filipe M. Vieira[2], Daniel F. Macedo[2], Alex B. Vieira[1]

[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

{cristiane.rodrigues,leonardocosta}@ice.ufjf.br, alex.borges@ufjf.edu.br

{mmvieira,lfvieira,damacedo}@dcc.ufmg.br

Abstract. *The decoupling between the control and data planes, the main idea from Software Defined Networks (SDNs), opens a new horizon to perform traditional tasks such as load balancing. In this sense, we present an SDN architecture in which different policies may be used to balance loads on Web servers. In this work we simulate system behavior when using three policies: random, round-robin style, and one that consider the Web server's workload. Our results show that for traditional Web loads, where there are several small requests, the policies do not differ significantly. For loads equivalent to Web 2.0, such as pictures and large files, the policy that considers the load presents an improvement of up to 11% in response time. Surprisingly, for heavy loads, which is becoming the most common load, the load-based policy has the worst result. In this case, the reaction time required by the Web application and SDN is much lower than the data collection time on Web servers.*

Resumo. *A separação entre camadas de controle e dados, ideia base de Redes Definidas por Software, abre um novo horizonte para realizar tarefas tradicionais como o balanceamento de carga. Nesse sentido, apresentamos uma arquitetura SDN na qual diferentes políticas podem ser utilizadas para realizar balanceamento de carga em servidores Web. Neste trabalho simulamos o comportamento do sistema ao se utilizar uma política aleatória, uma estilo round-robin e uma que leva em consideração a carga de trabalho dos servidores Web. Nossos resultados mostram que para cargas tradicionais Web, onde há várias requisições de pequeno tamanho para cada uma delas, as políticas não apresentam diferenças significativas. Para cargas equivalentes a Web 2.0, como fotos e arquivos grandes, a política que leva em consideração a carga apresenta uma melhora de até 11% no tempo de resposta. Surpreendentemente, para cargas pesadas, cada vez mais comuns, a política que leva em consideração a carga apresenta o pior resultado entre as três políticas consideradas. Nesse caso, o tempo de reação requisitado pela aplicação Web e SDN é muito inferior ao tempo de coleta de dados de carga nos servidores Web.*

1. Introdução

O balanceamento de carga tornou-se um mecanismo importante para serviços em redes de computadores. Com a inclusão de novas tecnologias, a Internet vem se tornando algo complexo, onde serviços devem ser capazes de lidar com milhares de requisições simultaneamente. Para suportar grandes demandas e manter um tempo de resposta suficientemente rápido para os clientes, é necessário que os recursos da rede sejam bem utilizados. Resumindo, cada servidor ou equipamento deve processar uma quantidade de trabalho compatível com sua capacidade.

Atualmente, uma determinada configuração de serviços em redes consiste em um balanceador de carga que recebe requisições recém chegadas e as distribui para múltiplos servidores pertencentes a uma rede. Comumente, esses balanceadores são sistemas (*hardware*/*software*) altamente especializados para determinado tipo de serviço. Por serem altamente especializados, esses balanceadores são equipamentos muito caros (facilmente superando 50 mil dólares), possuem regras rígidas de funcionamento e necessitam de administradores capacitados para sua operação [Uppal and Brandon 2010].

Redes Definidas por Software (SDN) são um novo paradigma que vem mudando a forma de como criar, modificar e gerenciar redes de computadores [Feamster et al. 2014]. Essa nova abordagem tem atraído a atenção tanto da indústria quanto da comunidade acadêmica. Com ela, diversos serviços de rede estão sendo repensados, de forma a torná-los mais flexíveis. O balanceamento de carga pode ser implementado empregando conceitos de SDN.

A separação entre camadas de controle e dados, ideia base de Redes Definidas por Software, abre um novo horizonte para realizar tarefas tradicionais como o balanceamento de carga. Por exemplo, podemos criar um balanceador de carga onde as regras de balanceamento são definidas de acordo com as várias aplicações existentes. Por utilizar regras adaptadas a cada aplicação, a distribuição de tráfego poderá ser mais eficiente do que empregarmos um balanceador proprietário. Além disso, por empregar *hardware commodity*, a solução SDN poderá ter um custo menor.

Na literatura, encontramos alguns esforços na direção de propor balanceamento de carga utilizando Redes Definidas por Software [Sherwood et al. 2009, Handigol et al. 2009, Uppal and Brandon 2010, Wang et al. 2011, Ragalatha et al. 2013, Zhou et al. 2014]. Os trabalhos citados propõem o balanceamento de carga através da construção de uma arquitetura SDN com o padrão OpenFlow, porém nem todos eles avaliam o comportamento do balanceador proposto. E os poucos que realizam tal avaliação, o fazem em ambiente muito específico, sem se preocupar com o perfil de carga a qual serão submetidos os serviços a serem balanceados.

Considerando esse contexto, nós propomos, neste trabalho, um arcabouço de balanceamento de carga em SDN no qual diferentes políticas podem ser utilizadas para realizar tal tarefa. Nós avaliamos o funcionamento desse arcabouço de balanceamento de carga utilizando um serviço *Web* como foco, e mostramos os benefícios alcançados e as limitações quando o balanceador tem que lidar com diferentes perfis de carga.

De uma forma mais detalhada, avaliamos o comportamento do sistema ao se utilizar três políticas de balanceamento diferentes: uma política aleatória, uma estilo *round-robin* e uma que leva em consideração a carga de trabalho dos servidores *Web*. Nós

também variamos a carga de trabalho, refletindo três cenários comuns a servidores *Web*.

Nossos resultados mostram que, para cargas tradicionais *Web* onde há várias requisições de pequeno tamanho, as políticas não apresentam diferenças significativas entre si. Nesse caso, as requisições são pequenas e de rápida resposta, e assim não há tempo suficiente para que políticas sofisticadas de balanceamento de carga convirjam para um resultado melhor que uma abordagem simples, como a *round-robin*. Para cargas equivalentes a *Web* 2.0, como *upload/download* de fotos e arquivos grandes, a política que leva em consideração a carga dos servidores *Web* apresenta uma melhora de até 11% no tempo de resposta, quando comparada às demais políticas. Surpreendentemente, para cargas *Web* pesadas, cada vez mais comuns na Internet, a política que leva em consideração a carga apresenta o pior resultado entre as três políticas consideradas. Nesse caso, o tempo de reação requisitado pela aplicação *Web* é muito inferior ao tempo de atualização dos dados de carga nos servidores *Web*. Assim, rajadas de requisições são direcionadas a um mesmo servidor que, erroneamente, foi interpretado como o menos sobrecarregado pelo controlador SDN.

O restante desse artigo está organizado como segue: na Seção 2 mostramos o estado da arte com relação às propostas de balanceamento de carga em SDN. Na seção 3 apresentamos nossa proposta e discutimos as decisões de projeto do nosso arcabouço de balanceamento de carga. As seções 4 e 5 mostram nossa metodologia de avaliação e os resultados obtidos com nosso balanceador de carga. Finalmente, a seção 6 apresenta nossas conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O balanceamento em redes de computadores é uma técnica usada para distribuir a carga de trabalho entre vários enlaces ou computadores [Uppal and Brandon 2010]. Desta forma, mais tarefas poderão ser realizadas na mesma quantidade de tempo e, em geral, todas as requisições poderão ser respondidas em menos tempo. Além disso, o balanceamento de carga traz vantagens como o aumento da escalabilidade, pois poderá existir flexibilidade em adicionar mais recursos à rede. Isso acontece de forma rápida e transparente aos usuários finais. Traz ainda como vantagem o aumento do desempenho pela utilização dos recursos de forma inteligente e maior disponibilidade, já que se um dos servidores falhar, a carga poderá ser redistribuída aos outros servidores existentes sem comprometer o processamento das requisições [Korner and Kao 2012].

Balanceamento de carga é um tema de pesquisa atrativo, uma vez que um balanceador dedicado pode se tornar um ponto de falha e congestionamento [Wang et al. 2011]. Além disso, algumas das soluções existentes são específicas para determinados serviços, o que as tornam pouco flexíveis com relação à visão da rede como um todo. Atualmente, diversos trabalhos em balanceamento de carga têm empregado os conceitos de SDN.

Grande parte da atenção em SDN tem sido voltada para o padrão OpenFlow [Guedes et al. 2012]. O OpenFlow é um dos principais protocolos relacionados a SDN e, de fato, é o que possibilita a implementação desse novo paradigma. Com a padronização do OpenFlow, vendedores e operadores de rede começam a dar uma importância maior a ele, reforçando por consequência o paradigma SDN. Por exemplo, em 2011, empresas como Facebook, Yahoo, Google e Microsoft formaram a Open Network-

ing Foundation (ONF). Essa organização tem como objetivo promover a tecnologia OpenFlow, trazendo para o mercado normas e soluções SDN [ONF 2014].

Alguns trabalhos recentes em balanceamento de carga apresentam propostas baseadas em SDN, entretanto não apresentam uma avaliação aprofundada das suas propostas [Handigol et al. 2009, Uppal and Brandon 2010, Wang et al. 2011, Ragalatha et al. 2013, Zhou et al. 2014]. De fato, alguns destes são trabalhos em desenvolvimento. Praticamente todos compartilham de semelhanças nas arquiteturas propostas: usam padrão OpenFlow, controlador baseado em NOX, e um esquema de reescrita tanto de cabeçalho de pacotes quanto da tabela de fluxos do switch OpenFlow.

Em alguns casos, o balanceamento de carga é efeito colateral da flexibilidade em se adicionar recursos de forma transparente à rede [Handigol et al. 2009]. Nesse trabalho, os autores desenvolveram um esquema que monitora a realidade da rede (recursos disponíveis) e distribui a carga de trabalho entre esses serviços. Para realizar essa distribuição, existem módulos específicos que monitoram os servidores existentes e seu estado. Esses módulos ativamente modificam as regras no controlador para realizar a distribuição de carga.

Outros trabalhos procuram realizar um balanceamento dinâmico de carga. Por exemplo, Ragalatha et al. [Ragalatha et al. 2013] implementam um balanceador onde a carga de trabalho é calculada e distribuída entre os servidores existentes em tempo de execução. Wang et al. [Wang et al. 2011] fazem o balanceamento dinâmico instalando regras curingas nos *switches*. Eles propõem algoritmos para computar essas regras e ajustá-las de acordo com mudanças no sistema. Isso ocorre sem quebras nas conexões já existentes.

Korner et al. [Korner and Kao 2012] também se preocupam com balanceamento de carga dinâmico. Porém, os autores tratam a rede de forma diferente dos trabalhos discutidos anteriormente. Nesse caso, o controlador NOX é utilizado juntamente com o FlowVisor, que divide os recursos de rede e utiliza um controlador diferente para cada divisão. Assim, cada fatia da rede pode ter seu próprio balanceamento, independente das demais. Isso provê maneiras de alocar recursos e balancear diferentes serviços, independentes um do outro.

Apesar dos trabalhos apontados mostrarem estratégias diversas para realizar balanceamento de carga em SDN, não ficam claros os benefícios alcançados. Alguns deles não realizam uma avaliação consistente, pois utilizam cargas simplistas quando comparados a cenários mais próximos à realidade, como os descritos em [Jeon et al. 2012, Polfliet et al. 2012, Bovenzi et al. 2011]. Dessa forma, esses trabalhos apresentam resultados que podem não ser conclusivos.

Assim, destacamos as seguintes contribuições de nosso trabalho: primeiro, apresentamos uma maneira simples e flexível para realizar balanceamento de carga em SDN. Nós discutimos a sua implementação e mostramos soluções para capturar informações da rede, sem quebrar a premissa de divisão entre dados e controle. Segundo, nós apresentamos uma avaliação do mecanismo implementado. Nós mostramos cenários próximos à realidade e mostramos as limitações existentes para balanceamento de carga em SDN.

3. O mecanismo de balanceamento de carga proposto

A proposta do presente trabalho é a criação e avaliação do balanceamento de carga em uma rede SDN com OpenFlow. Nossa proposta se concentra no balanceamento de cargas *Web*. Porém, a arquitetura proposta é flexível, permitindo a coexistência de serviços e políticas de balanceamento, e podendo ser empregada em outros serviços além da *Web*.

A Figura 1 apresenta a arquitetura proposta nesse trabalho. Os servidores hospedam os servidores *Web* da arquitetura. Esses servidores se ligam ao mundo externo por um único endereço IP, ao qual também iremos nos referir como endereço IP externo do serviço. Quando um cliente faz uma nova requisição para o endereço IP externo do serviço, a requisição é encaminhada para um *switch* OpenFlow. Esse *switch* tem regras específicas e encaminha a carga a um dos servidores do sistema, balanceando assim a carga total de serviço. As regras de balanceamento que o *switch* executa são definidas pelo controlador OpenFlow.

Mais detalhadamente, o funcionamento do sistema ocorre como descrito a seguir: quando o primeiro pacote de uma nova solicitação chega ao *switch*, ele é repassado ao controlador. O controlador, após a escolha de qual servidor do conjunto de servidores que respondem ao serviço irá atender a requisição, cria uma entrada na tabela de fluxo do *switch*, com as ações de encaminhar o pacote para a porta de saída específica. Além disso, é feita a reescrita dos campos de endereços MAC e IP de destino, substituindo os endereços MAC e IP do endereço de serviço pelos endereços MAC e IP do servidor escolhido. Nesse momento, o controlador também instala regras para permitir o fluxo de retorno do servidor para o cliente. Neste caso serão modificados os campos de origem do pacote. Para a resposta à requisição, os endereços IP e MAC de origem do pacote, que são os do servidor, serão substituídos pelos endereços IP externos de serviço. Com isso, o cliente não sabe para qual servidor foi enviada sua requisição e de qual servidor ele obteve resposta.

Nesse trabalho, utilizamos três políticas para realizar o balanceamento das requisições, a saber: Round-Robin, Aleatória e Baseada na carga. As políticas foram implementadas na linguagem Python. As políticas implementadas para o balanceamento de carga são detalhadas a seguir:

1. **Aleatória:** O controlador escolhe aleatoriamente para qual servidor enviar a solicitação. Essa escolha é uniformemente distribuída entre os servidores existentes.
2. **Round-Robin:** Para cada fluxo novo, um servidor é escolhido alternativamente para receber a solicitação do cliente, a partir do primeiro servidor. Assim, as requisições são divididas uniformemente entre todos os servidores.
3. **Baseada na carga:** Essa política envia a solicitação para o servidor com a menor carga média de CPU. Essa carga média é calculada pelo sistema operacional através da média do último minuto de uso da CPU. A carga é obtida no arquivo de sistema /PROC/LOADAVG de cada servidor Linux. Os servidores enviam as suas cargas para a máquina onde se encontra o controlador, através de comunicação estabelecida via Sockets Python, por uma rede externa à rede SDN.

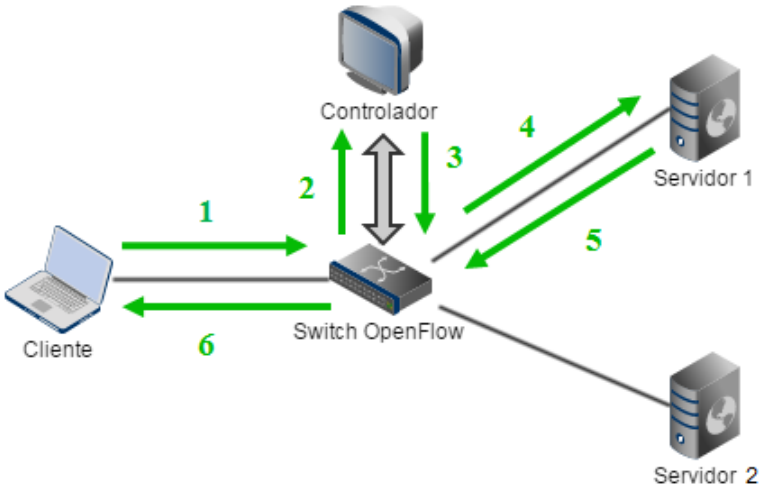


Figura 1. Arquitetura da rede proposta: (1) A requisição do cliente é enviada ao *switch*; (2) Como não há entrada especificada na tabela de fluxos, o primeiro pacote da requisição é repassado ao controlador; (3) O controlador escolhe o servidor de destino, cria uma entrada correspondente na tabela de fluxos e devolve o pacote ao *switch*; (4) A requisição é enviada ao servidor escolhido; (5) A requisição é atendida pelo servidor, que envia a resposta de volta ao *switch*; (6) A resposta da requisição é enviada ao cliente.

4. Metodologia de Avaliação

4.1. Ambiente de testes

A Figura 1 mostra a arquitetura SDN criada para a realização dos testes do esquema de balanceamento de carga proposto. A arquitetura foi criada dentro de uma máquina física, e é composta por máquinas virtuais e um *switch* virtual. Utilizamos duas máquinas virtuais como servidores *Web* de nossa arquitetura. Esses servidores têm duas interfaces de rede cada. Uma das interfaces está conectada à rede definida por *software*.

Para realizar a comunicação direta entre os servidores e controlador, sem violar o princípio básico de separação dos planos de controle e dados existente em Redes Definidas por Software, uma rede externa à SDN foi criada. Essa rede externa foi criada utilizando uma segunda placa de rede contida na máquina física, tornando possível a comunicação direta com os servidores externamente à rede SDN.

Utilizamos o Open vSwitch (OvS) [Pfaff et al. 2009] como elemento de chaveamento de nossa arquitetura. O OvS é um *switch* virtual multi-camadas com suporte OpenFlow e de código aberto, construído para funcionar em ambientes virtualizados. Ele suporta a criação de VLANS e GRE tunneling, além de fornecer conectividade entre as máquinas virtuais e as interfaces físicas. Através da interface física, é criada uma ponte para que possam ser criadas novas interfaces virtuais que se conectarão às máquinas servidoras.

O controlador SDN empregado foi o POX [POX 2014]. Ele foi instalado no mesmo equipamento onde se encontra o *switch* OpenFlow. No controlador foram im-

plementadas as três políticas de escalonamento avaliadas nesse trabalho. Conforme mencionamos, o controlador recebe informações sobre os servidores em uma interface à parte da SDN. Por essa interface, o controlador recebe a carga de CPU de cada servidor.

Os cliente de testes e os dois servidores *Web* foram instanciados como três máquinas virtuais. Essas máquinas estão em execução dentro da máquina física onde se encontra o *switch* OpenFlow, e são gerenciadas através do VirtualBox. Os servidores possuem sistema operacional Linux Ubuntu Server 12.04, com 1 GB de memória. Em cada servidor foi instalado o servidor *Web* Apache versão 2.2. O Apache, que é o servidor *Web* mais utilizado atualmente, foi escolhido por ser de fácil uso e por atender bem aos requisitos da arquitetura. Não houve necessidade de configurações adicionais no Apache. Já a máquina cliente possui sistema operacional Ubuntu 14.04, com 512 MB de memória. Nela foi instalada a ferramenta que foi responsável por enviar as requisições ou cargas: o Httperf [Mosberger and Jin 1998].

4.2. Cargas de trabalho

Para enviar as cargas, a partir da máquina cliente, foi utilizada a ferramenta Httperf. O Httperf é uma ferramenta que tem como propósito medir o desempenho de um servidor *Web*. Ela gera diversas cargas de trabalho HTTP a partir de parâmetros pré-definidos. Ao final de sua execução, o Httperf sumariza as estatísticas gerais de desempenho do servidor avaliado, como o tempo total de conexão, por exemplo.

Os testes foram realizados em três cenários distintos. Em cada um deles utilizamos as três políticas de balanceamento de carga descritas na Seção 3. O objetivo dos testes é obter os tempos médios de conexão, através de experimentos, para diferentes taxas de chegada de requisições por segundo, para a posterior avaliação de desempenho dos algoritmos em cada situação. Definimos um experimento como sendo cada execução do Httperf com uma taxa de chegada pré-definida, que retornará um valor de tempo total de conexão. Esse experimento é executado 10 vezes para cada valor de taxa de chegada, e a média dos tempos de conexão é então calculada. As taxas de chegada são um conjunto de valores determinísticos, definidos como parâmetro no Httperf. Várias taxas foram utilizadas nos experimentos em cada cenário, porém as mais representativas em termos gráficos foram escolhidas para ilustrar o comportamento de cada política de balanceamento.

O primeiro cenário retrata uma carga *Web* leve. Intuitivamente, tentamos retratar o acesso a uma página comum da *Web* com textos e figuras pequenas [Bovenzi et al. 2011, Williams et al. 2005, Goseva-Popstojanova et al. 2004]. Nesse sentido, o acesso é feito a um pequeno arquivo HTML. Cada acesso a esse pequeno arquivo gera 20 requisições de 10 KB cada, totalizando 200 KB. Para cada experimento geramos 1.000 conexões, com taxas de chegada variando entre 100 e 1000 requisições por segundo.

O segundo cenário retrata uma carga típica *Web* 2.0. Nesse cenário, usuários enviam e recuperam de servidores *Web* arquivos nitidamente maiores que simples páginas. Por exemplo, fotos e arquivos de texto com cerca de 1 MB de dados [Jeon et al. 2012, Polfliet et al. 2012, Nagpurkar et al. 2008]. Assim, avaliamos o balanceador de carga utilizando requisições a um arquivo (i.e. JPG) de 1 MB. Novamente, cada experimento tem 1.000 conexões e as taxas de chegada variam entre 10 e 1.000.

Por fim, no terceiro cenário, testamos o ambiente com um arquivo de vídeo de 100

MB. Esse cenário de carga pesada é cada vez mais comum na Internet. Observamos que, cada vez mais, usuários Internet estão acessando vídeos de longa duração e com qualidade alta [Mitra et al. 2011, Summers et al. 2012]. Nesse cenário, cada experimento tem 100 conexões, cada uma com uma requisição pelo vídeo. Variamos as taxas de chegada entre 1 e 100 conexões por segundo.

Os resultados que apresentamos, como já dissemos, são valores médios de 10 execuções de cada experimento, com intervalos de confiança de 95%.

5. Resultados

Nesta parte do trabalho serão mostrados os resultados dos testes realizados para cada um dos cenários que foram citados na seção anterior. Os gráficos apresentam a relação entre o tempo médio de duração das conexões e a taxa de requisições enviadas pelo cliente. O propósito de analisar essa relação é verificar o impacto do aumento da taxa sobre o tempo de processamento das conexões.

5.1. Primeiro cenário

No primeiro cenário, os testes foram realizados com um arquivo HTML. Em cada um dos 10 experimentos são enviadas 1000 conexões com 20 requisições cada. A Figura 2 apresenta o tempo médio de resposta (e intervalo de confiança) a uma requisição, enquanto variamos a taxa de chegada nos servidores.

Para cargas leves, as três políticas de balanceamento apresentam resultados semelhantes entre si. De fato, para todas as taxas avaliadas, os intervalos de confiança das políticas apresentam interseção entre si. Intuitivamente, as cargas leves são atendidas muito rapidamente. Assim, qualquer um dos servidores que for delegado a atender uma nova requisição estará com sobra de recursos computacionais.

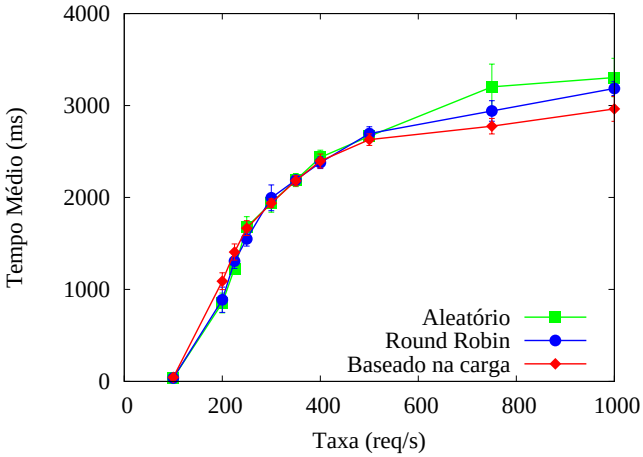


Figura 2. Tempo médio por taxa para o arquivo HTML

Com taxas mais elevadas (e.g. 500 a 750 req./s), há uma leve diferença entre as políticas. Nesse cenário, a política baseada na carga é estatisticamente diferente da abordagem aleatória. Mais precisamente, para taxas de 750 req./s, as abordagens baseada na carga, *round-robin* e aleatória apresentam tempos médios de resposta de 2775, 2941 e

3202 ms respectivamente. Para essa carga, as recusas de conexão pelo servidor *Web* são baixas para todas as políticas e variam entre 0,67% a 1,8%.

Com o aumento da carga, observamos também um aumento no percentual de requisições recusadas pelos servidores. Além disso, com uma taxa alta, a utilização dos servidores pode mudar rapidamente. Infelizmente, o mecanismo Linux que provê estatísticas de carga de um servidor apresenta médias do último minuto avaliado. Assim, um servidor, com baixa carga no último minuto, pode receber uma rajada de conexões. Isso faz com que o tempo de resposta médio e a taxa de recusas piorem, se comparados com taxas menores. De fato, como observamos na figura, para a taxa de 1.000 req./s, a política baseada na carga tem resultados equivalentes à política aleatória. Porém, a taxa de recusa de conexões chega a 7,5% para a política baseada na carga dos servidores, contra 0,65% da política *round-robin*.

5.2. Segundo cenário

No segundo cenário, os testes foram realizados com um arquivo JPG. Novamente, em cada um dos 10 experimentos são realizadas 1000 conexões, com uma requisição a cada arquivo.

De acordo com a Figura 3, para uma taxa baixa de requisições, todas as três políticas possuem resultados equivalentes. Nessa situação (taxa baixa de chegada), os servidores *Web* do sistema tem recursos suficientes para atender as requisições. Taxas baixas não geram rajadas a um único servidor (o que pode acontecer pela política aleatória e por carga).

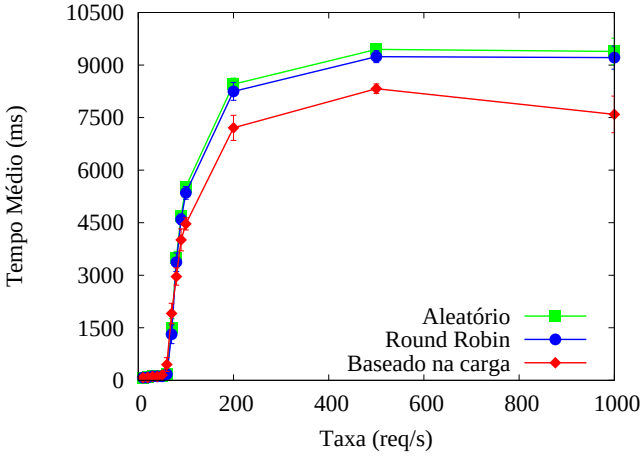


Figura 3. Tempo médio por taxa para o arquivo JPG

Para taxas a partir de 90 req./s, a política baseada na carga passa a ter melhor desempenho que as demais. Por exemplo, para uma taxa de 500 req./s, a política baseada na carga é 12% melhor que a aleatória. Mais detalhadamente, a Figura 4 apresenta a distribuição de probabilidade acumulada do tempo de resposta para taxas de chegada de 500 req./s e 20 execuções de experimentos. Por essa figura, fica claro verificar que as políticas Round-Robin e aleatória são equivalentes. Entretanto, a política baseada na carga pode apresentar respostas até 1s mais rápidas que as demais.

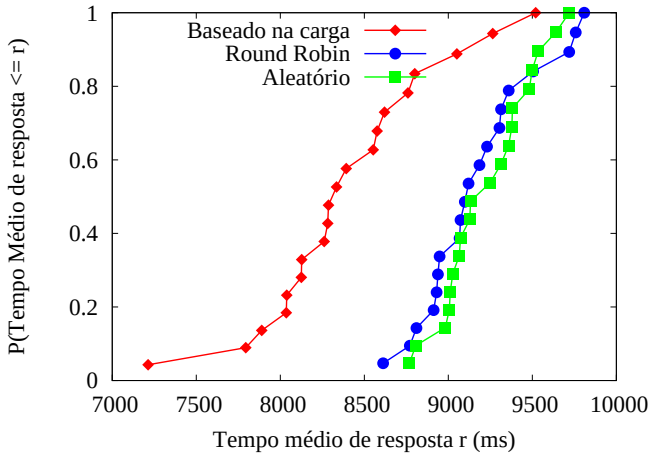


Figura 4. Tempo de Resposta para Taxas de Chegada de 500 req./s

Novamente, para taxas muito altas, observamos rajadas de requisições. Isso faz com que os servidores rejeitem novos clientes e aumentem assim a taxa de recusa. No cenário atual, o percentual de recusa é consideravelmente maior, quando comparado ao primeiro cenário. Por exemplo, para a política baseada na carga, uma taxa de 1000 req./s gerou 13,16% de recusas.

5.3. Terceiro cenário

O terceiro cenário de testes apresenta cargas pesadas a serem processadas pelo sistema. Nesse cenário, realizamos 100 conexões em cada execução do experimento. Cada conexão requisita um arquivo de vídeo de 100MB.

A Figura 5 apresenta os tempos médios de resposta para taxas de chegada variando de 1 a 100 req./s. Para praticamente todas as taxas avaliadas, as três políticas não são equivalentes. Porém, ao contrário dos outros dois cenários, a política baseada na carga tende a ser a pior.

Nesse cenário, as requisições demoram a ser processadas pelo servidor *Web*. Vídeos grandes são executados em minutos, ao contrário dos poucos segundos observados nas cargas dos dois cenários anteriores. Os servidores *Web* utilizados têm recursos limitados, e assim, saturam-se rapidamente. Como consequência, tão logo algumas poucas requisições consumam os recursos dos servidores *Web*, qualquer tentativa de balanceamento de carga se mostra ineficiente.

Além disso, nesse cenário, a letargia em atualizar o valor da carga média de trabalho da CPU pelo sistema operacional de cada um dos servidores gera resultados ainda piores que o esperado. Como, no início dos experimentos, o servidor que apresenta o menor valor médio de carga de trabalho será escolhido, toda nova requisição será direcionada para esse servidor. Porém, como uma requisição de vídeo demora a ser processada, ela ficará impactando o servidor até que o sistema operacional informe ao controlador da rede uma atualização de sua carga de trabalho. Vale ressaltar que a atualização do valor da carga da CPU é lenta, pois este valor refere-se à média de cargas do último minuto. Como, neste caso, apenas um servidor será responsável por atender a quase todas as requisições, e o outro servidor praticamente não terá nenhuma requisição para atender,

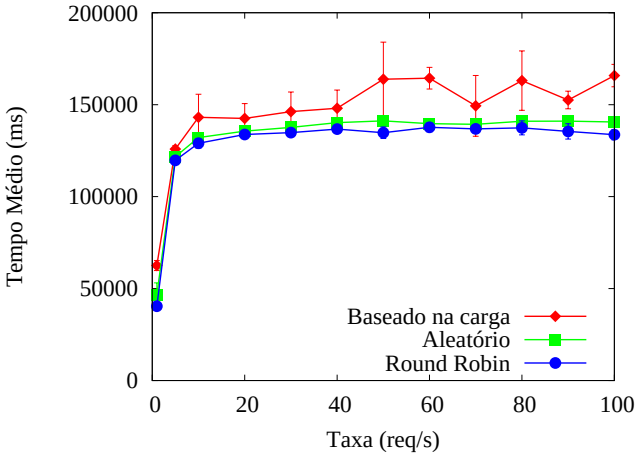


Figura 5. Tempo médio por taxa para o arquivo de vídeo

os tempos médios de resposta ficarão maiores para esse cenário.

6. Considerações Finais

A consolidação do paradigma de Redes Definidas por Software vem mudando a forma de gerenciar e projetar as redes de computadores, permitindo que elas possam evoluir e melhorar mais rapidamente em relação ao que ocorre atualmente. Dessa forma, com SDN foi possível criar um ambiente de teste para estudar um tema que tem um importante papel nas redes de computadores: o balanceamento de carga.

Este trabalho propõe e avalia o uso de SDN para o balanceamento de carga. Foram avaliadas três políticas de balanceamento de carga, a saber: política aleatória, política *round-robin* e uma política ciente da carga dos servidores de aplicação. As soluções propostas foram avaliadas em três cenários *Web*.

No primeiro cenário, equivalente a um cenário *Web* típico onde páginas são requisitadas a um servidor, as três políticas de balanceamento apresentam desempenhos semelhantes. As requisições consomem poucos recursos e são atendidas rapidamente. Assim, qualquer servidor pode atender requisições com recursos suficientes. Para cargas equivalentes a *Web* 2.0, como fotos e arquivos grandes, a política que leva em consideração a carga apresenta uma melhora de até 11% no tempo de resposta. Isso representa respostas aos clientes *Web* até 1 segundo mais rápidas. Surpreendentemente, para cargas pesadas, cada vez mais comuns, a política que leva em consideração a carga apresenta o pior resultado entre as três políticas consideradas. Nesse caso, o tempo de reação requisitado pela aplicação *Web* é muito inferior ao tempo de atualização de dados da carga nos servidores *Web*. Essa letargia por parte do sistema operacional faz com que rajadas de requisições sejam encaminhadas a um único servidor, aumentando também o tempo médio de resposta das requisições.

A arquitetura proposta pode ter melhores resultados a partir de duas frentes: primeiro, podemos refinar o esquema de medição de carga disponível no sistema operacional do servidor. Esse esquema deve ser rápido e reagir a novas cargas tão rápido quanto mudam as condições da rede. Segundo, podemos mesclar as políticas de balanceamento de carga e adicionar mecanismos para evitar rajadas.

Também pretendemos avaliar a proposta, e novas políticas de balanceamento de carga, em um ambiente de produção. Nesse sentido, estamos criando um arcabouço SDN equivalente ao proposto nesse trabalho, utilizando os recursos existentes em um *campus* universitário. Podemos apontar também como trabalho futuro extensões ao balanceamento de carga, como por exemplo, criar políticas de economia de energia entre os servidores existentes.

Agradecimentos

Os autores agradecem o apoio do CNPq, CAPES e da FAPEMIG.

Referências

Bovenzi, A., Cotroneo, D., Pietrantuono, R., and Russo, S. (2011). Workload characterization for software aging analysis. In *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pages 240–249.

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.

Goseva-Popstojanova, K., Mazimdar, S., and Singh, A. D. (2004). Empirical study of session-based workload and reliability for web servers. In *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pages 403–414.

Guedes, D., Vieira, L. F. M., Vieira, M. M., Rodrigues, H., and Nunes, R. V. (2012). Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. In *Minicurso do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2012*, Ouro Preto, Brasil.

Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N., and Johari, R. (2009). Plug-n-serve: load-balancing web traffic using openflow. In *Proceedings of demo at ACM SIGCOMM*.

Jeon, M., Kim, Y., Hwang, J., Lee, J., and Seo, E. (2012). Workload characterization and performance implications of large-scale blog servers. *ACM Transactions on the Web*, 6(4).

Korner, M. and Kao, O. (2012). Multiple service load-balancing with openflow. In *Proceedings of the IEEE 13th Conference on High Performance Switching and Routing*. IEEE Publishers, Belgrado, Sérvia.

Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D., and Mahanti, A. (2011). Characterizing web-based video sharing workloads. *ACM Transactions on the Web*, 5(2).

Mosberger, D. and Jin, T. (1998). httperf: A tool for measuring web server performance. In *Proceedings of the 1998 Internet Server Performance Workshop*, volume 26, pages 31–37.

Nagpurkar, P., Horn, W., Gopalakrishnan, U., Dubey, N., Jann, J., and Pattnaik, P. (2008). Workload characterization of selected jee-based web 2.0 applications. In *Proceedings of IEEE International Symposium on Workload Characterization 2008*, pages 109–118.

ONF (2014). What is onf? Disponível em: https://www.opennetworking.org /images/stories/downloads/about/onf-what-why.pdf. Acessado em: Novembro de 2014.

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M., and Shenker, S. (2009). Extending networking into the virtualization layer. In *8th ACM Workshop on Hot Topics inNetworks (HotNets-VIII). New York, NY, USA*.

Polfliet, S., Ryckbosch, F., and Eeckhout, L. (2012). Studying hardware and software trade-offs for a real-life web 2.0 workload. In *ICPE'12 - Proceedings of the 3rd Joint WOSP/SIPEW International Conference on Performance Engineering*, pages 181–192.

POX (2014). Pox wiki. Disponível em: https://openflow.stanford.edu/display/ONL/POX+Wiki. Acessado em: Novembro de 2014.

Ragalatha, P., Challa, M., and K, S. K. (2013). Design and implementation of dynamic load balancer on openflow enabled sdns. In *Research Paper in International Organization of Scientific Research Journal of Engineering (IOSRJEN), ISSN: 2250-3021*, volume 3, Issue 8, pages 32–41.

Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. (2009). Flowvisor: A network virtualization layer. In *Technical Report Openflow-tr-2009-1, OpenFlow*. Stanford University.

Summers, J., Brecht, T., Eager, D., and Wong, B. (2012). Methodologies for generating http streaming video workloads to evaluate web server performance. In *ACM International Conference Proceeding Series*.

Uppal, H. and Brandon, D. (2010). Openflow based load balancing. In *Proceedings of CSE561: Networking Project Report*. University of Washington, Spring.

Wang, R., Butnariu, D., and Rexford, J. (2011). Openflow-based server load balancing gone wild. In *Hot-ICE'11 Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*. USENIX Association Berkeley, CA, USA.

Williams, A., Arlitt, M., Williamson, C., and Barker, K. (2005). Web workload characterization: Ten years later. In *Web Content Delivery*, volume 2 of *Web Information Systems Engineering and Internet Technologies Book Series*, pages 3–21. Springer US.

Zhou, Y., Ruan, L., Xiao, L., and Liu, R. (2014). A method for load balancing based on software defined network. In *Advanced Science and Technology Letters*, volume 45, pages 43–48.

Mecanismo de Controle de Acesso ao Meio para Redes Sem Fio com Rádios Full-Duplex*

Marcel William Rocha da Silva[1], **José Ferreira de Rezende**[2]

[1] DCC – IM – Universidade Federal Rural do Rio de Janeiro (UFRRJ)

[2]COPPE – Universidade Federal do Rio de Janeiro (UFRJ)

marcelsilva@ufrrj.br, rezende@land.ufrj.br

Abstract. *Recently, the construction of full-duplex radios became a reality due to advances obtained in the self-interference cancellation area. However, the potential performance gains from this new type of radio could be explored only by using a specific medium access control mechanism. This work aims to propose and evaluate the performance of a new media access protocol to wireless networks using full-duplex radios. The proposed mechanism uses the basis of the IEEE 802.11 protocol, striving to efficiently explore opportunities for bidirectional communication provided by the use of full-duplex radios.*

Resumo. *Há pouco tempo, a construção de rádios full-duplex se tornou realidade devido aos avanços obtidos na área de cancelamento de auto-interferência. Entretanto, os potenciais ganhos de desempenho obtidos com este novo tipo de rádio somente são explorados com o uso de um mecanismo de controle de acesso ao meio específico. Este trabalho tem como objetivo propor e avaliar o desempenho de um novo protocolo de acesso ao meio para redes sem fio que utilizam rádios full-duplex. O mecanismo proposto utiliza como base o protocolo IEEE 802.11 e busca explorar de maneira eficiente oportunidades de comunicação bidirecional disponibilizadas pela utilização dos rádios full-duplex.*

1. Introdução

Avanços recentes nas técnicas de cancelamento de auto-interferência (*self-interference cancellation*) permitiram a implementação de rádios full-duplex [Choi et al. 2010, Jain et al. 2011, Choi and Lim 2012, Sen et al. 2012]. Com isso, tornou-se viável a construção de rádios que transmitem e recebem simultaneamente em um único canal. Teoricamente, esses rádios permitem dobrar a capacidade das redes sem fio atuais. No entanto, esses ganhos dependem drasticamente de um protocolo de controle de acesso ao meio (MAC) que permita aproveitar ao máximo as oportunidades de transmissão bidirecional e seja capaz de lidar com problemas de justiça no compartilhamento da banda, do terminal escondido e exposto, os quais são ainda maiores nas redes com comunicação bidirecional.

Trabalhos atuais na literatura propõem protocolos MAC que lidam com esses problemas [Singh et al. 2011, Sahai et al. 2011, Zhou et al. 2013, Xie and Zhang 2014]. No entanto, percebe-se uma grande complexidade envolvida nos mecanismos de acesso ao

*Este trabalho recebeu recursos da FAPERJ.

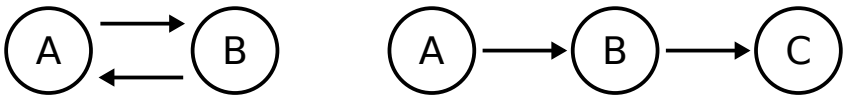
meio propostos nesses trabalhos, o que pode inviabilizar a implementação desses protocolos. A necessidade desses protocolos em manter um conhecimento da vizinhança e estatísticas associadas a cada um dos vizinhos, adiciona uma alta complexidade aos nós sem fio. Além disso, a maioria das propostas requer que o transmissor busque na fila de quadros a serem transmitidos aquele a ser transmitido a cada instante, diferente daquele quadro que ocupa a cabeça da fila.

Neste trabalho, propomos um protocolo MAC full-duplex, baseado no mecanismo DCF do 802.11, que permite aproveitar as oportunidades em enlaces bidirecionais simétricos, garantindo um certo nível de justiça no compartilhamento do meio. Na concepção desse protocolo foi adotado um compromisso entre desempenho e simplicidade para garantir a sua implementabilidade em dispositivos comerciais de baixo custo. Desta forma, este protocolo MAC full-duplex, denominado FDMAC, usa uma máquina de estados simplificada e não requer a manutenção de estados da vizinhança. Essa última propriedade garante o bom funcionamento do protocolo mesmo em caso de mobilidade dos nós. A avaliação de desempenho do protocolo proposto foi realizada através de simulações, as quais mostram os potenciais ganhos do emprego de rádios full-duplex em conjunto com um protocolo MAC de acesso aleatório.

O texto a seguir está estruturado da seguinte forma. Na Seção 2, será apresentado um breve resumo dos trabalhos relacionados existentes. O protocolo MAC proposto neste trabalho será descrito na Seção 3. Em seguida, a Seção 4 apresenta o ambiente de simulação e analisa os resultados obtidos na avaliação de desempenho realizada. Por fim, a Seção 5 apresenta as conclusões e as perspectivas deste trabalho.

2. Trabalhos relacionados

Recentemente, vários protocolos MAC full-duplex foram propostos na literatura [Singh et al. 2011, Sahai et al. 2011, Zhou et al. 2013, Xie and Zhang 2014]. Do ponto de vista do protocolo MAC, enlaces full-duplex podem operar em dois modos como descrito em [Xie and Zhang 2014]: modo bidirecional e modo de encaminhamento [Xie and Zhang 2014]. O primeiro modo permite a um par de nós transmitirem um ao outro simultaneamente, enquanto o segundo modo permite que um nó receptor reencaminhe quadros a outro nó vizinho ao mesmo tempo (Figura 1).



(a) bidirecional (b) encaminhamento

Figura 1. Modos de operação full-duplex.

Em [Singh et al. 2011], esses dois modos são classificados em enlaces duais simétricos e assimétricos, respectivamente. O uso destes dois modos de operação permite um melhor aproveitamento das oportunidades de transmissão simultânea. Uma oportunidade assimétrica é mais frequente que a bidirecional pois basta que o receptor (nó B) tenha quadros a serem transmitidos a um de seus vizinhos (nó C). No entanto, essa oportunidade assimétrica se torna real apenas quando a interferência causada no nó C pelo transmissor primário seja inferior a um determinado limiar. Assim, a detecção dessas oportunidades exige um grande aumento na complexidade do protocolo MAC [Singh et al. 2011].

Outra questão com relação à complexidade diz respeito à necessidade de uma busca por quadros fora da ordem na fila de transmissão. Ou seja, para o aproveitamento dessas oportunidades é necessário que o nó seja capaz de percorrer a sua fila de quadros de transmissão para detectar a presença de um quadro destinado a um de seus vizinhos. Esse tipo de mecanismo deve levar em conta a possibilidade de quadros na cabeça da fila sofrerem maiores atrasos no acesso ao meio ou até mesmo sofrerem de inanição (*starvation*). Por exemplo, em [Singh et al. 2011], os autores propõem, para a seleção do quadro a ser transmitido, o uso de uma lista dos vizinhos ranqueada pela taxa de sucesso das últimas x (=10) transmissões. Além disso, a escolha desse quadro deve ser feita rapidamente de modo a não afetar a transmissão simultânea com um quadro em recepção.

Em [Xie and Zhang 2014], os autores apresentam um modelo analítico para quantificar o ganho em capacidade fornecido pelo uso de rádios full-duplex em relação ao uso de rádios half-duplex. Os resultados obtidos por uma análise assimptótica em redes de larga-escala mostram que os efeitos do reuso espacial e da disputa assíncrona pelo meio prejudicam drasticamente os ganhos dos rádios full-duplex. Como conclusões desse trabalho, algumas diretrizes para a concepção de protocolos MAC full-duplex são fornecidas.

O protocolo FD-MAC, proposto em [Sahai et al. 2011], foi concebido especialmente para redes infra-estruturadas. Diversos mecanismos são propostos nesse protocolo para aumentar as possibilidades de uso do full-duplex. Esses mecanismos vão desde a troca de informação entre dois nós sobre os contadores a serem usados nos *backoffs* até a descoberta da topologia para a formação de enlaces duais assimétricos, passando pela busca de quadros fora da ordem na fila de transmissão.

Alguns desses protocolos propõem o uso do tom de ocupado (*busy tone*) proposto originalmente no protocolo BTMA [Tobagi and Kleinrock 1975]. O tom de ocupado, quando transmitido pelo receptor, permite evitar os problemas do terminal escondido e exposto. Diferentemente do BTMA, onde o tom de ocupado é transmitido em outro canal, nos protocolos MAC full-duplex, onde ocorre o cancelamento da auto-interferência, o tom de ocupado é transmitido no mesmo canal. Quando o receptor tem um quadro a ser transmitido, ele precisa apenas transmitir o tom de ocupado ao final da transmissão caso a duração do quadro sendo transmitido seja inferior ao do quadro em recepção. No caso do receptor não ter nenhum a quadro a ser transmitido simultaneamente, ele transmite o tom de ocupado durante toda a recepção do quadro. Em todos os casos, a recepção é protegida de potenciais terminais escondidos. Apesar das aparentes vantagens trazidas pelo uso do tom de ocupado, o seu uso é considerado por alguns autores como um acréscimo de complexidade que não traz ganhos significativos, sem contar com os gastos de energia incorridos na sua utilização e o aumento da interferência co-canal gerada em uma rede de larga-escala.

Um outro problema bastante relevante na concepção de novos protocolos MAC full-duplex está relacionado à justiça no compartilhamento da banda pelos múltiplos nós. Um grande número de trabalhos apontam para o compartilhamento injusto fornecido pelo mecanismo DCF do 802.11, o qual está relacionado a diversos fatores tais como à captura do meio por parte de alguns nós, às condições topológicas e ao uso do espaçamento entre quadros `EIFS`. Desta forma, a maioria das propostas de protocolos MAC full-duplex atacam o problema da justiça, definindo mecanismos que levam a um melhor comparti-

lhamento do meio.

3. Proposta

O protocolo MAC para rádios full-duplex proposto neste trabalho, denominado FDMAC, consiste em modificações ao mecanismo DCF do padrão IEEE 802.11 para permitir o aproveitamento das oportunidades de comunicação. O FDMAC utiliza apenas as oportunidades simétricas, não realizando o reordenamento da fila de transmissão. Assim, ao iniciar a recepção de um quadro, cada nó espera a chegada completa do cabeçalho e então verifica se ele tem um quadro na cabeça da fila a ser transmitido ao transmissor do quadro em recepção. Nesse caso, ele transmite o quadro simultaneamente. Caso contrário, apenas a transmissão em um sentido é realizada. Este funcionamento é exemplificado na Figura 2, onde vemos ao longo do tempo a transmissão simultânea de dois quadros entre os nós A e B.

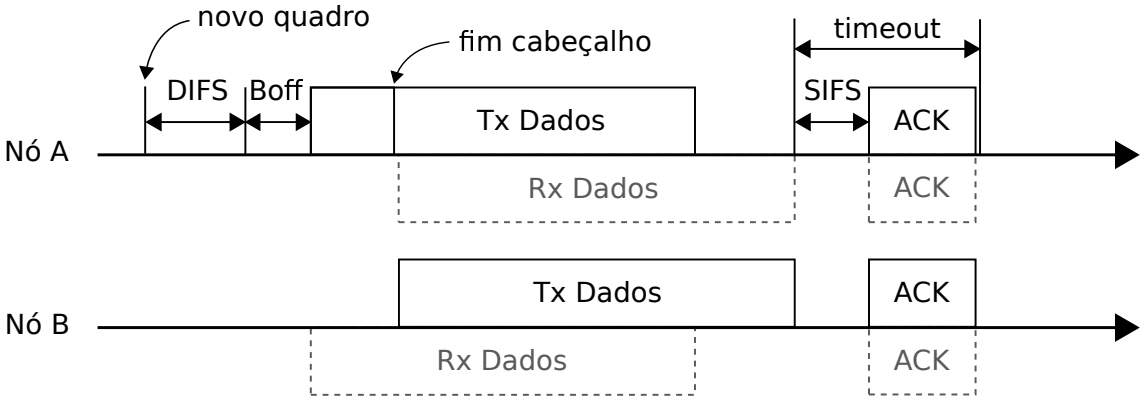


Figura 2. Exemplo de comunicação bidirecional.

Na transmissão simultânea (modo bidirecional), ao término da recepção de um quadro, o nó apenas pode enviar um reconhecimento (`ACK`) quando a sua transmissão tiver sido finalizada. Assim, em caso de transmissões simultâneas, os nós podem ter que adiar a transmissão de ACKs. Esse procedimento exige que o transmissor inicialize o temporizador usado para detectar a não recepção do `ACK` em diferentes instantes de tempo. Quando ele estiver apenas transmitindo, ele deve inicializar o temporizador ao final da transmissão do quadro (caso do nó B na Figura 2). No caso em que ele estiver ainda em modo de recepção (caso do nó A na Figura 2), ele deve inicializar este temporizador apenas ao final da recepção do quadro. Em ambos os casos, o temporizador será inicializado com um valor igual ao tempo de espaçamento entre quadros `SIFS` adicionado ao tempo de transmissão do `ACK` e ao tempo máximo de propagação ida e volta, como mostrado na Figura 2. Desta forma, pode ser notado que quando duas transmissões acontecem simultaneamente, uma sincronização no envio dos ACKs ocorre, pois o envio de um dos reconhecimentos será sempre adiado.

Como explicado na seção anterior, os protocolos MAC full-duplex derivados do DCF ressaltam ainda mais o problema de injustiça no compartilhamento da banda. Para minimizar o problema, alguns mecanismos foram acrescidos ao protocolo FDMAC. Como descrito anteriormente, a sincronização dos ACKs entre um par de nós em uma comunicação bidirecional, apesar de corretamente decodificados por ambos os nós, é

percebido como uma colisão por um terceiro nó que se encontra na vizinhança de ambos os nós em comunicação. De acordo com a especificação do DCF do padrão IEEE 802.11 [802.11 1999], sempre que uma colisão ocorre e o nó não consegue decodificar uma transmissão, ele utiliza o espaçamento entre quadros EIFS antes de iniciar a decrementar o *backoff* para acessar o meio. Este espaçamento EIFS possui valor correspondente à soma de: SIFS, DIFS e o tempo de transmissão do ACK. Como este valor é superior ao valor de DIFS, que normalmente é utilizado antes do *backoff* aleatório, o par de comunicação que obteve o meio pela primeira vez continua transmitindo, caso ele sempre tenha quadros a transmitir, sem permitir que um outro nó capture o meio. Esse comportamento leva a uma grande injustiça na utilização da banda.

Para resolver esse problema, a seguinte solução foi empregada no FDMAC. Sempre que o receptor se sincroniza para a decodificação de um quadro, ele guarda o instante de tempo (t_{inicio}) em que essa sincronização ocorreu. Ao final da recepção (t_{fim}), se o nó não consegue decodificar o quadro corretamente, ele estima o tempo em que ficou nesse estado ($t_{colisao} = t_{fim} - t_{inicio}$). Se este tempo $t_{colisao}$ estimado é próximo ao tempo de transmissão de um ACK (T_{ack}), ele considera que houve uma colisão de ACKs e espera DIFS, ao invés de EIFS, antes de uma nova tentativa de acessar ao meio. Como o EIFS foi concebido exatamente para se evitar a colisão de ACKs [802.11 1999], a não utilização desse espaçamento não traria nenhum malefício, uma vez que a colisão já ocorreu.

A Figura 3 apresenta um diagrama de tempo onde uma colisão de ACKs em uma transmissão bidirecional entre os nós A e B é percebida por um terceiro nó C. Neste cenário, inicialmente o nó C percebe uma colisão (representadas em cinza na figura) com duração longa, a qual é maior que o tempo de transmissão de um ACK. Neste caso, a ação padrão de esperar EIFS antes de decrementar o *backoff* é executada. Após um intervalo de tempo SIFS, uma nova colisão é percebida por C, desta vez uma colisão de ACKs. Neste caso, logo após o nó C ao perceber que a duração da colisão foi equivalente ao tempo de transmissão de um ACK, uma espera de DIFS é realizada antes de decrementar o *backoff* aleatório. Como isso, pode-se perceber que o nó C consegue disputar o acesso ao meio com os nós A e B em igualdade de condições. Isso é importante para que algum par de comunicação não utilize exclusivamente o meio em detrimento aos demais nós sem fio presentes na região.

Fica evidente que para o funcionamento correto do mecanismo apresentado anteriormente é importante a identificação de uma colisão de ACKs apenas através de sua duração, já que os ACKs não poderão ser decodificados. Logo, a percepção do meio ocupado e em colisão por um intervalo de tempo curto provocado por quadros de dados, não pode ser confundida com o meio ocupado por uma colisão de ACKs. De sorte, o tempo de transmissão de quadros de dados que encapsulam datagramas IP de qualquer tamanho é bem superior ao tempo necessário para transmitir um ACK [Xiao and Rosdahl 2002].

4. Ambiente de Simulação e Resultados Numéricos

Para avaliar o desempenho da proposta foram realizados experimentos com o simulador ns-2 [NS-2 2007]. Para a implementação do protocolo FDMAC nesse simulador a eventos discretos, utilizou-se como base o seu módulo de camada MAC 802.11 original, onde diversas modificações foram realizadas, dentre elas: a remoção dos quadros de controle RTS/CTS; a remoção das restrições que impediam as transmissões full-duplex; o acrés-

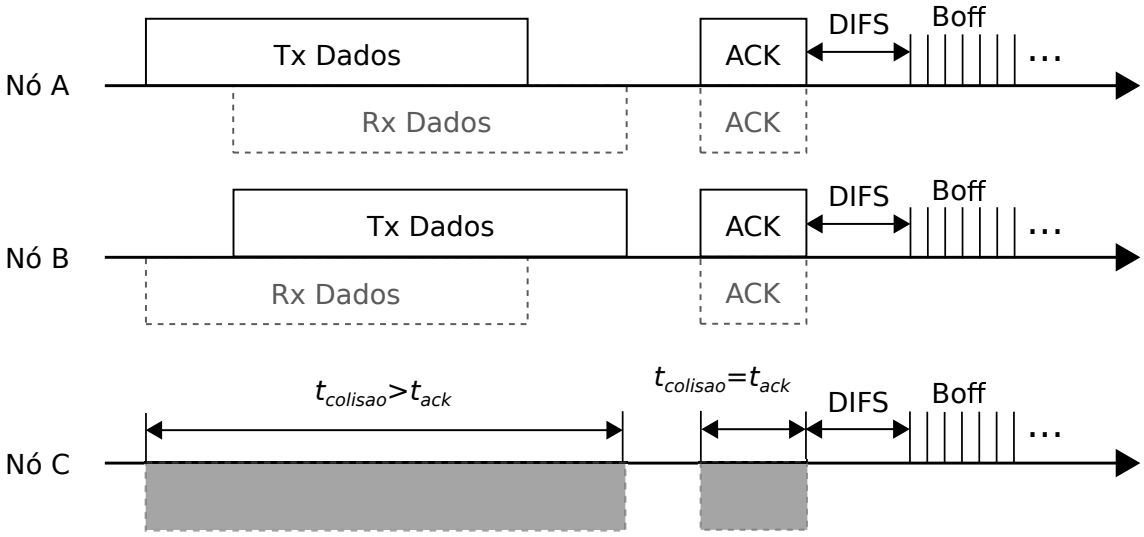


Figura 3. FDMAC em cenário de colisão de ACKs bidirecionais.

cimo de um novo estado na máquina de estados do protocolo para o tratamento simultâneo de espera e envio de ACKs; o acréscimo de um temporizador para o processamento ao final da recepção do cabeçalho MAC; a modificação dos temporizadores para a sincronização no envio dos ACKs bidirecionais, a implementação do mecanismo de justiça na colisão dos ACKs explicado na Seção 3, e a inclusão da verificação de restrições adicionais para o correto funcionamento do simulador.

Além disso, outra modificação importante adicionada ao ns-2 foi a representação da influência entre as transmissões dos nós sem fio através de um modelo de interferência mais realista do que aquele presente no ns-2. Neste novo modelo, a interferência acumulada durante a recepção dos quadros é contabilizada, e somente são recebidos corretamente quadros cuja SINR ficou acima de um limite mínimo definido. Os valores utilizados neste modelo foram: SINR mínima de 10 dB e potência de transmissão dos rádios de 10 dBm. Associado à este modelo de interferência utilizou-se o modelo de propagação *log-distance* [Rappaport 2001], com parâmetro $\alpha = 5$, para estimar a perda de potência no percurso. Com estes parâmetros o alcance de recepção correta de um quadro é de aproximadamente 95 metros para um taxa de transmissão de dados de 11 Mbps. Tais características equivalem à um ambiente urbano *indoor*.

As simulações realizadas foram divididas em dois conjuntos de experimentos distintos, os quais serão apresentados em detalhe a seguir.

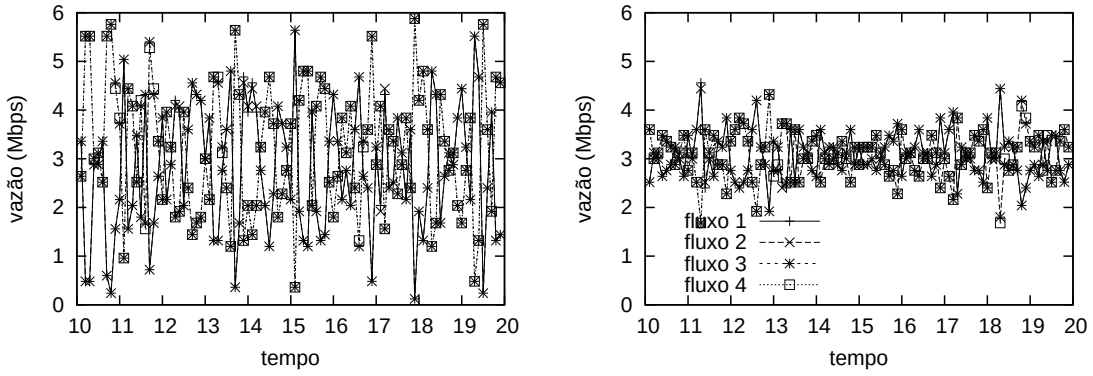
4.1. Avaliação do Mecanismo de Correção do `EIFS` do Protocolo FDMAC

O primeiro conjunto foi executado em um cenário ilustrativo simples, no qual o objetivo era avaliar o funcionamento do mecanismo de justiça na colisão dos ACKs proposto para o protocolo FDMAC. Para tal, dois pares de nós sem fio foram posicionados próximos uns aos outros, com alcance suficiente para a decodificação dos quadros. Apesar de ilustrativo, este é um tipo de arranjo muito comum em ambientes urbanos *indoor*, onde residências ou escritórios próximos possuem pontos de acesso sem fio com poucos clientes cada.

Cada par (a, b) de nós sem fio utilizado na simulação gera um fluxo de dados

bidirecional, ou seja, um fluxo do nó a para o nó b e outro do nó b para o nó a. Estes fluxos de dados consistem de pacotes de 1500 bytes que são gerados a uma taxa constante (CBR) e suficientemente grande para garantir que os *buffers* de camada MAC sempre possuam pelo menos um quadro a ser transmitido, ou seja, operem na saturação.

Nos gráficos das Figuras 4 e 5 são apresentadas as vazões de cada fluxo de dados ao longo do tempo calculadas com intervalos de tempo de $0,1$ e $0,5$ segundos, respectivamente. Em ambos os casos, são apresentadas as vazões instantâneas dos fluxos com e sem o mecanismo de correção do `EIFS`. Apesar de simples, estas simulações mostram bem o mecanismo em funcionamento. Nos casos onde o mecanismo não está habilitado (Figuras 4(a) e 5(a)), pode se perceber uma maior variabilidade das vazões dos fluxos ao longo do tempo. A causa desta variação é a injustiça na disputa pelo acesso ao meio em curtas escalas de tempo. Quando um par sem fio acessa o meio de comunicação, ele tem maior probabilidade de conseguir acessar novamente nas próximas tentativas, já que estes nós sem fio aguardam apenas `DIFS`+*backoff* para transmitir novamente. O outro par possui uma chance menor de acessar o meio de comunicação, já que após a colisão dos ACKs bidirecionais vai aguardar `EIFS`+*backoff* antes de transmitir. Eventualmente, o par que acessa o meio escolhe um valor de *backoff* grande e permite que o outro par transmita. Entretanto, por períodos de tempo significativos, da ordem de centenas de milissegundos, um dos pares transmite enquanto o outro fica impossibilitado de acessar o meio.



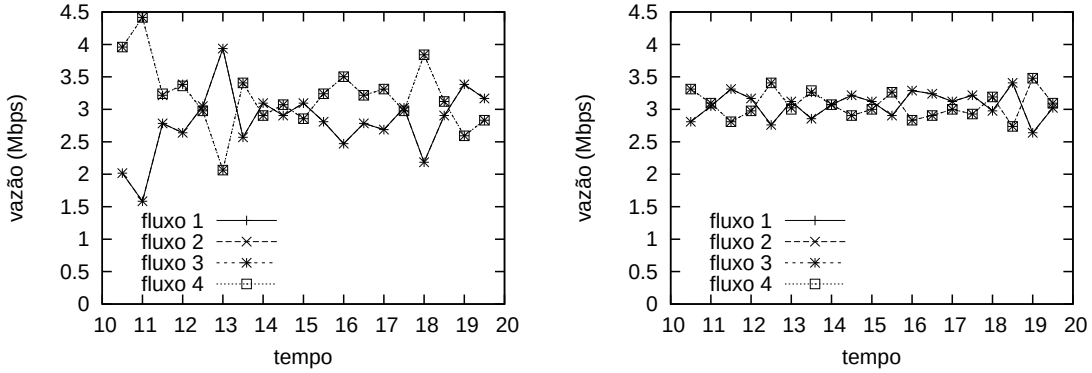
(a) sem correção do `EIFS` (b) com correção do `EIFS`

Figura 4. Vazão de cada fluxo a intervalo de 0.1 segundos.

Os efeitos desta injustiça em curtas escalas de tempo pode ser especialmente prejudicial para o desempenho das aplicações. A demora para conseguir acesso ao meio sem fio fará com que as filas de transmissão fiquem cheias, o que pode ocasionar perdas de pacotes, aumento do atraso e da variação do atraso. Desta forma, aplicações do tipo tempo-real ficam prejudicadas e também aplicações que utilizam transferência confiável de dados com o protocolo TCP, já que as perdas de pacotes na fila podem prejudicar o mecanismo de controle de congestionamento. Voltaremos a discutir este assunto no próximo conjunto de experimentos.

4.2. Avaliação do Desempenho do Protocolo FDMAC em Cenários Aleatórios

O segundo conjunto de experimentos consistiu na distribuição aleatória de pares de nós sem fio em uma área quadrada. Neste posicionamento aleatório, garantiu-se apenas que



(a) sem correção do EIFS (b) com correção do EIFS

Figura 5. Vazão de cada fluxo a intervalo de 0.5 segundos.

os nós sem fio de um mesmo par estivessem dentro do alcance de recepção um do outro. Foram realizadas simulações com uma quantidade de pares de nós sem fio variando de 1 até 11 em áreas quadradas de 100, 500 e 1000 metros de lado. Além disso, dois tipos de tráfego distintos foram utilizados nas simulações: tráfego CBR, semelhante ao utilizado na seção anterior, e tráfego TCP de longa-duração, usado na transferência de grandes arquivos, durante toda a simulação. O objetivo era avaliar o desempenho do protocolo proposto em diferentes situações de carga e disposição dos nós de forma a permitir o reuso espacial do canal.

O desempenho para os dois tipos de camada MAC são apresentados nos gráficos: o algoritmo DCF **802.11 original** e o algoritmo **FDMAC proposto**, como descrito na Seção 3. Para os resultados do FDMAC dois conjuntos de simulações foram realizadas: com e sem o mecanismo de correção do EIFS avaliado na seção anterior.

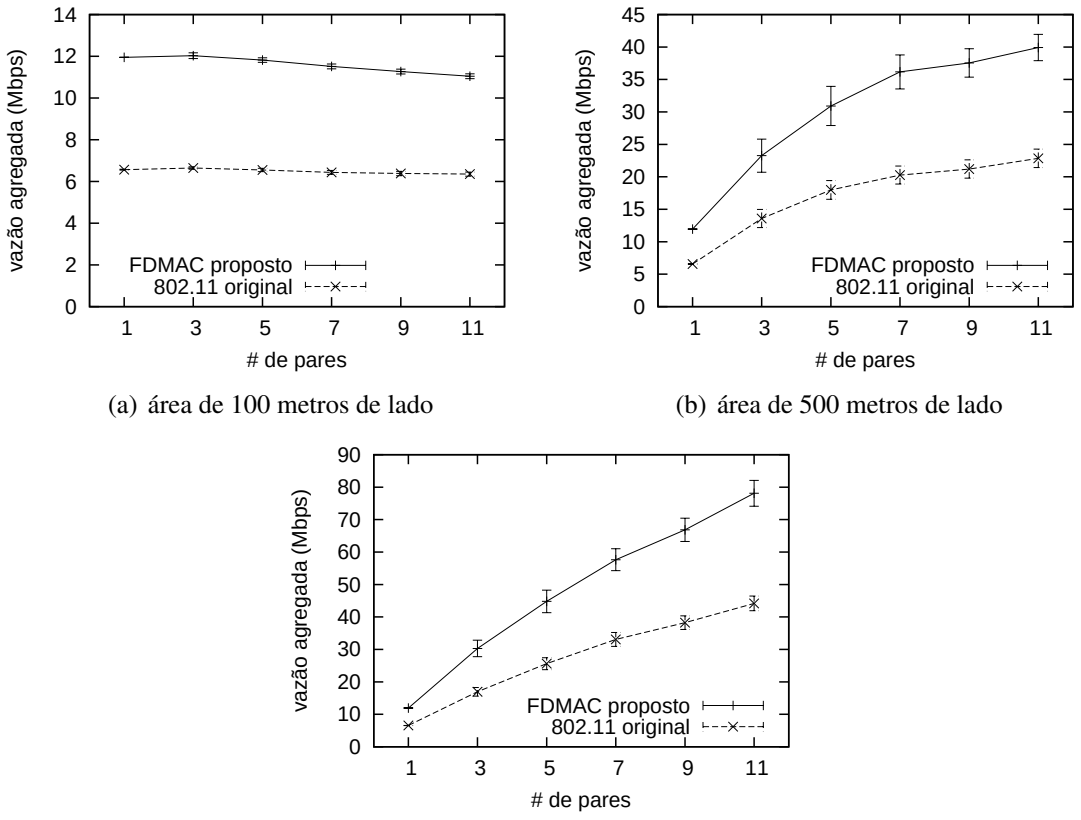
Para cada possível configuração, ou seja, cada combinação de quantidade de pares de nós e de tamanho do lado da área quadrada, foram executadas 30 rodadas de simulação com 100 segundos de duração. Cada rodada utilizou um posicionamento aleatório diferente dos pares de nós sem fio, como mencionado anteriormente. Em todos os gráficos apresentados a seguir, os pontos representam os valores médios das 30 rodadas de simulação executadas para cada configuração, e as barras de erro correspondem a intervalos de confiança de 95%.

As métricas de desempenho avaliadas nestas simulações foram a **vazão agregada** e o **índice de justiça**. A vazão agregada é o somatório das vazões individuais dos fluxos de dados existentes na simulação, e fornece uma medida do quanto da capacidade do meio de comunicação está sendo utilizada pelos pares comunicantes. Já o índice de justiça (*fairness index*) [Jain et al. 1984] é uma métrica, com valores no intervalo $[0, 1]$, a qual mede o nível de proximidade de um conjunto de amostras. O cálculo do valor do índice de justiça é dado pela equação $\frac{(\sum x_i)^2}{n \sum \left(x_i^2\right)}$, a qual fornece a justiça entre os n valores x_i.

Neste novo conjunto de experimentos, o desempenho do protocolo FDMAC proposto com ou sem o mecanismo de correção do EIFS habilitado forneceu resultados semelhantes, com exceção dos cenários com tráfego TCP de longa duração e área quadrada de lado 100 metros. Isso já era esperado, uma vez que as métricas avaliadas nestas simu-

lações medem o desempenho médio dos protocolos a longo prazo, mascarando os efeitos prejudiciais da injustiça no acesso ao meio em curtas escalas de tempo. Desta forma, por questão de melhorar a apresentação dos resultados, os casos onde o mecanismo de correção do `EIFS` não gerou diferenças significativas foram omitidos. Já os resultados no caso onde a diferença de desempenho é pronunciada estão incluídos e serão discutidos a seguir.

Os gráficos da Figura 6 apresentam a vazão agregada em Mbps para os protocolos 802.11 original e FDMAC em cenários de tráfego CBR. De acordo com os resultados, pode-se perceber que a utilização do protocolo FDMAC forneceu ganhos de desempenho significativos na vazão agregada, independente da quantidade de pares de comunicação e também do tamanho da área. Tais ganhos em grande parte dos cenários são próximos do ganho máximo teórico de 2 vezes que seria possível pela adoção de rádios full-duplex. Este resultado se justifica já que os experimentos criam um cenário favorável ao mecanismo FDMAC, onde todos os nós sem fio possuem quadros a ser transmitidos e podem aproveitar as oportunidades de transmissão bidirecional.



(a) área de 100 metros de lado

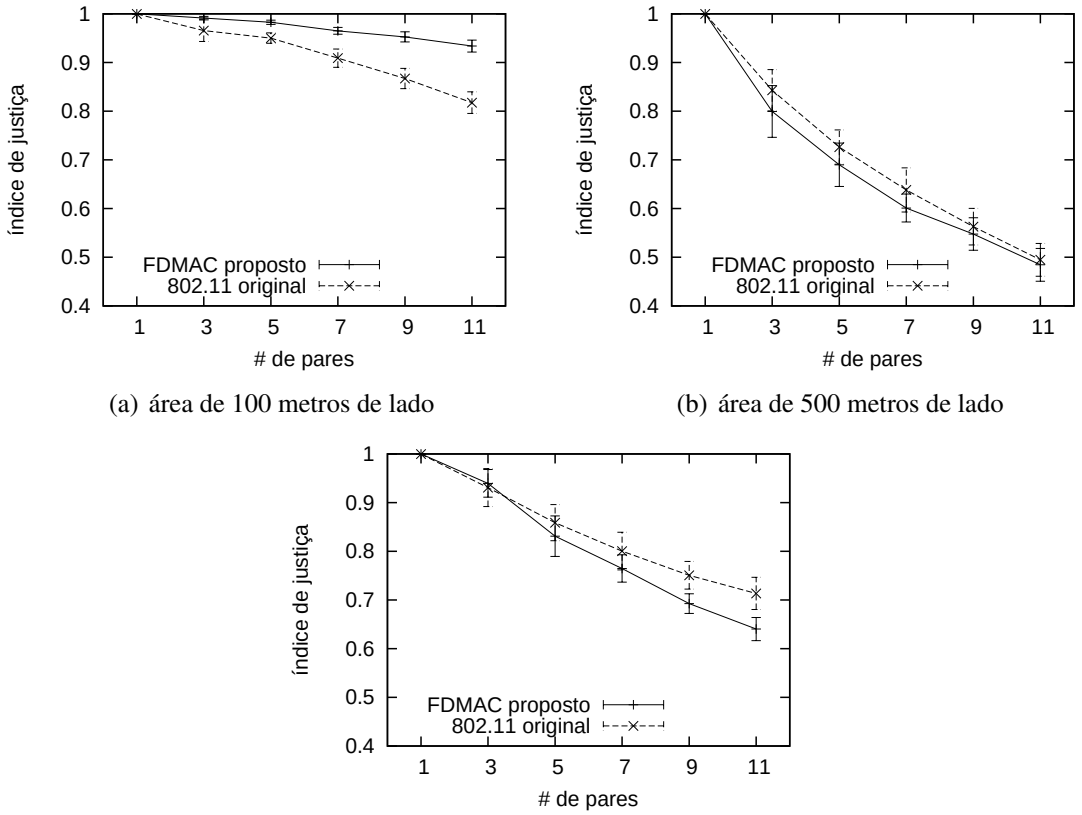
(b) área de 500 metros de lado

(c) área de 1000 metros de lado

Figura 6. Vazão agregada com tráfego CBR e com mecanismo de correção do `EIFS`.

Ainda a respeito dos experimentos utilizando o tráfego CBR, a Figura 7 apresenta os resultados obtidos para o índice de justiça. De acordo com estes gráficos, pode-se perceber que o protocolo FDMAC proposto forneceu ganhos na justiça entre as vazões dos fluxos quando operando em cenários densos (Figura 7(a)) onde, por conseguinte, não

existe reuso espacial. Isso se explica em parte pelo melhor aproveitamento das oportunidades de transmissão, pois agora quando um nó sem fio ganha o acesso ao meio, uma oportunidade de transmissão bidirecional é gerada também para o seu par de comunicação. Desta forma, os fluxos de dados bidirecionais em um mesmo par de comunicação tendem a obter vazões próximas. Além disso, este acoplamento gerado pelas transmissões bidirecionais faz com que a disputa pelo acesso ao meio aconteça entre os pares de comunicação, ao invés de entre nós sem fio independentes. Logo, a alternância no acesso ao meio ocorre em uma escala de tempo menor, o que faz com que a espera no acesso ao meio seja menor.



(a) área de 100 metros de lado

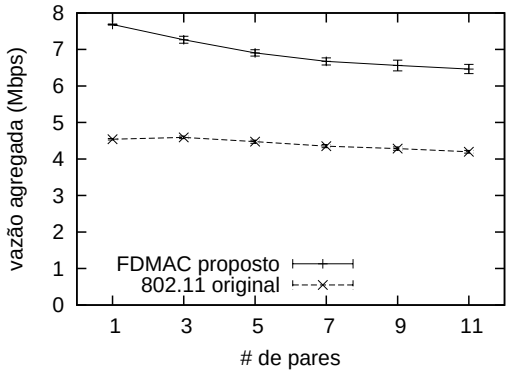
(b) área de 500 metros de lado

(c) área de 1000 metros de lado

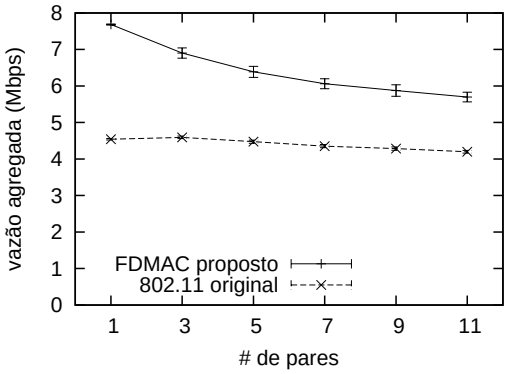
Figura 7. Índice de justiça com tráfego CBR e com mecanismo de correção do `EIFS`.

Os próximos conjuntos de gráficos da Figura 8 apresentam a vazão agregada nos experimentos com fluxos bidirecionais TCP. Para estes novos cenários de tráfego, percebe-se que os ganhos obtidos com o uso do FDMAC não são tão pronunciados quanto aqueles obtidos no caso do tráfego CBR. Isto se justifica, pois o protocolo TCP envia, além dos dados da aplicação, mensagens de confirmação de camada de transporte (ACKs TCP). Com isso, as transmissões bidirecionais, frequentemente, consistem na combinação de um segmento de dados TCP e um ACK TCP. Esta característica do funcionamento do protocolo TCP faz com que nem todas as oportunidades de transmissão bidirecional sejam aproveitadas para o envio de dados da aplicação e torna o aproveitamento da capacidade do meio de comunicação menos eficiente.

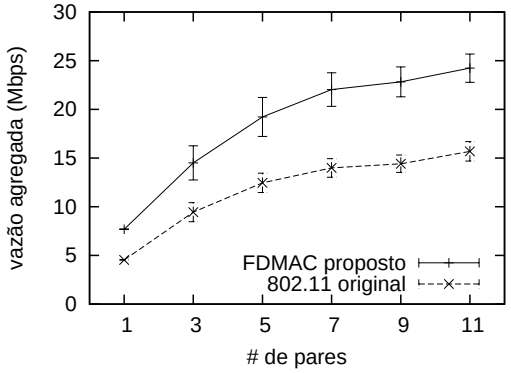
Além do problema mencionado no parágrafo anterior, a utilização do tráfego TCP de longa-duração torna a comunicação mais sensível à perdas de mensagens quando comparada ao tráfego CBR. Isto se deve principalmente ao mecanismo de controle de congestionamento do protocolo TCP, que precisa retransmitir mensagens quando estas são perdidas pela camada MAC. Este efeito fica mais evidente quando analisamos os resultados para o cenário denso, com área quadrada de 100 metros de lado (Figuras 8(a) e 8(b)). Neste caso, o aumento do número de pares de comunicação causa uma redução da vazão agregada, devido ao aumento da quantidade de colisões entre transmissões de pares distintos e, consequentemente, a perda de quadros na camada MAC. Entretanto, vale ressaltar que, ainda assim, em todos os experimentos foram obtidos ganhos de desempenho significativos pelo uso do protocolo FDMAC.



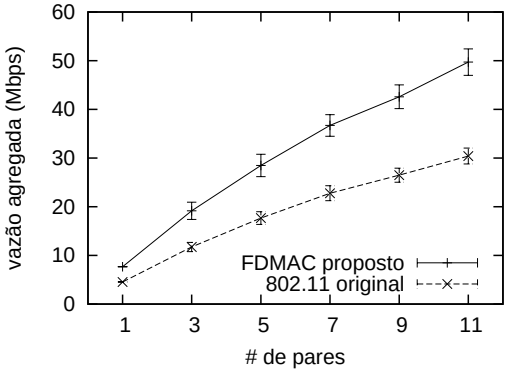
(a) área de 100 metros de lado, sem mecanismo de correção do EIFS



(b) área de 100 metros de lado, com mecanismo de correção do EIFS



(c) área de 500 metros de lado, com mecanismo de correção do EIFS

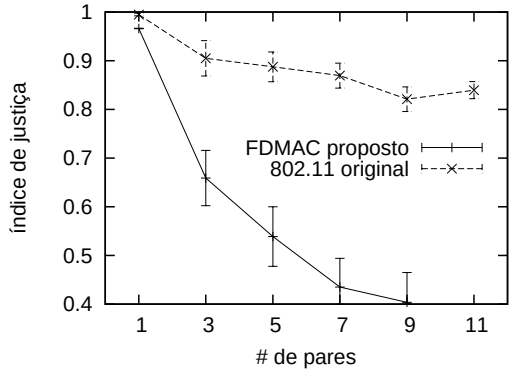


(d) área de 1000 metros de lado, com mecanismo de correção do EIFS

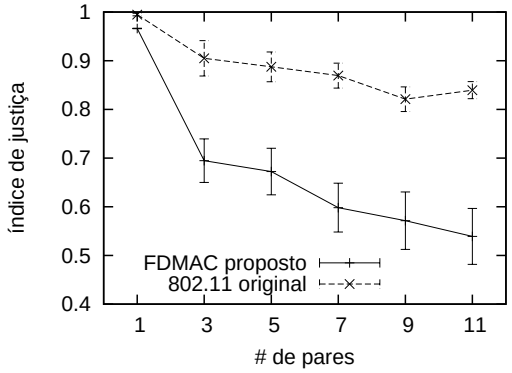
Figura 8. Vazão agregada com tráfego TCP.

O próximo conjunto de resultados apresentado na Figura 9 mostra o índice de justiça entre as vazões individuais nos experimentos com fluxos TCP. Nestes resultados, vemos que a justiça entre os fluxos com o uso do FDMAC é inferior quando comparada ao 802.11 original. Nos cenários de área quadrada com 100 metros de lado (Figura 9(a)) a piora na justiça é ainda mais expressiva. Entretanto, ao comparar os gráficos das Figuras 9(a) e 9(b), pode-se notar que o uso do mecanismo de correção do EIFS ameniza um pouco a injustiça. Isso ocorre, pois a maior frequência na alternância do acesso ao meio torna mais regular o intervalo de envio das mensagens de dados e ACK do protocolo

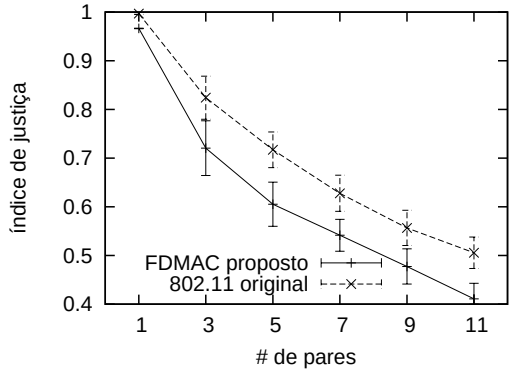
TCP por um único nó sem fio e, consequentemente, melhora a justiça entre as vazões individuais.



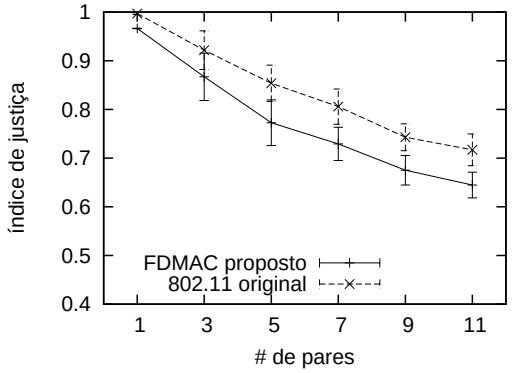
(a) área de 100 metros de lado, sem mecanismo de correção do EIFS



(b) área de 100 metros de lado, com mecanismo de correção do EIFS



(c) área de 500 metros de lado, com mecanismo de correção do EIFS



(d) área de 1000 metros de lado, com mecanismo de correção do EIFS

Figura 9. Índice de justiça com tráfego TCP.

5. Conclusões e Trabalhos Futuros

Os avanços obtidos na área de cancelamento de auto-interferência permitiram uma quebra de paradigma nas comunicações sem fio: a realização de comunicação full-duplex em uma única faixa de frequências. Esta técnica tem o potencial de ampliar a capacidade das comunicações sem fio em diversos cenários. Entretanto, muitos avanços ainda são necessários para que todo o potencial dos rádios full-duplex possa ser explorado na prática. Uma dos problemas existentes são os protocolos de camada MAC de acesso aleatório, como o utilizado pelas redes do padrão IEEE 802.11. Tal protocolo foi pensado inicialmente para o uso de rádios half-duplex e não consegue explorar as oportunidades de transmissão simultânea fornecidas pelos rádios full-duplex.

Neste trabalho, apresentamos um protocolo de camada MAC full-duplex simples, denominado FDMAC, que utiliza rádios full-duplex para explorar as oportunidades de comunicação bidirecional. No protocolo proposto, detectou-se a necessidade de modificar a ação padrão utilizada pelo 802.11 para reagir à colisões, que seria aguardar o tempo EIFS+*backoff* antes de transmitir novamente. Devido à natureza da comunicação bidirecional, a colisão entre quadros de ACK é muito provável, e seria injusto esperar um tempo

longo neste caso. O mecanismo auxiliar do FDMAC proposto identifica esta situação e toma uma ação mais justa, o que é importante para garantir justiça em curtas escalas de tempo, que poderiam prejudicar o desempenho das comunicações.

A avaliação de desempenho mostrou que, na maioria dos cenários, o protocolo FDMAC proposto consegue aumentar a vazão agregada sem comprometer a justiça entre as vazões dos fluxos de dados individuais. Este é um resultado muito satisfatório, que mostra que a utilização de protocolos de acesso ao meio aleatório em conjunto com rádios full-duplex é viável. Além disso o mecanismo de correção do `EIFS` se mostrou eficiente para solucionar problemas de justiça em curtas escalas de tempo e também melhorar a justiça do FDMAC em redes densas. Entretanto, alguns dos resultados obtidos comprovam o que foi constatado na literatura, que a justiça em protocolos full-duplex de acesso ao meio é um desafio a ser estudado.

Como trabalhos futuros pretende-se realizar novas avaliações de desempenho com o protocolo FDMAC. Dentre as avaliações pretendidas estão experimentos utilizando redes infra-estruturadas, redes de múltiplos saltos e experimentos envolvendo mobilidade dos nós sem fio. Para isso, será necessário estender o protocolo FDMAC para dar suporte ao modo de encaminhamento além do modo bidirecional como descrito na Seção2. Além disso, pretende-se explorar também questões relacionadas ao aprimoramento da justiça no acesso ao meio com o protocolo FDMAC.

Referências

802.11 (1999). Wireless lan medium access control (MAC) and physical layer (PHY) specifications. IEEE Standard.

Choi, J. I., Jain, M., Srinivasan, K., Levis, P., and Katti, S. (2010). Achieving single channel, full duplex wireless communication. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, pages 1–12, New York, NY, USA. ACM.

Choi, W. and Lim, H. (2012). Immediate acknowledgement for single-channel full-duplex wireless networks. In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pages 477–478.

Jain, M., Choi, J. I., Kim, T., Bharadia, D., Seth, S., Srinivasan, K., Levis, P., Katti, S., and Sinha, P. (2011). Practical, real-time, full duplex wireless. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 301–312, New York, NY, USA. ACM.

Jain, R., Chiu, D., and Hawe, W. (1984). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report Report TR-301, DEC Research.

NS-2 (2007). The network simulator - ns-2. `http://www.isi.edu/nsnam/ns/`. Último acesso em 15/12/2014.

Rappaport, T. (2001). *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

Sahai, A., Patel, G., and Sabharwal, A. (2011). Pushing the limits of full-duplex: Design and real-time implementation. *CoRR*, abs/1107.0607.

Sen, S., Choudhury, R., and Nelakuditi, S. (2012). CSMA/CN: Carrier sense multiple access with collision notification. *Networking, IEEE/ACM Transactions on*, 20(2):544–556.

Singh, N., Gunawardena, D., Proutiere, A., Radunovic, B., Balan, H., and Key, P. (2011). Efficient and fair MAC for wireless networks with self-interference cancellation. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2011 International Symposium on*, pages 94–101.

Tobagi, F. and Kleinrock, L. (1975). Packet switching in radio channels: Part II–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution. *Communications, IEEE Transactions on*, 23(12):1417–1433.

Xiao, Y. and Rosdahl, J. (2002). Throughput and delay limits of IEEE 802.11. *Communications Letters, IEEE*, 6(8):355–357.

Xie, X. and Zhang, X. (2014). Does full-duplex double the capacity of wireless networks? In *INFOCOM, 2014 Proceedings IEEE*, pages 253–261.

Zhou, W., Srinivasan, K., and Sinha, P. (2013). RCTC: Rapid concurrent transmission coordination in full duplex wireless networks. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–10.

Um Sistema de Decisão de Acesso para Conectividade Contínua em Redes Heterogêneas sem Fio

Robson Melo, Renan Polisciuc, Michele Nogueira, Aldri Santos

[1]Núcleo de Redes Sem Fio e Redes Avançadas – UFPR – PR – Brasil

{rgmelo,rsp12,michele,aldri}@inf.ufpr.br

Abstract. *The overlay of several wireless technologies, such as Wi-Fi and LTE, in a geographic region has increased with the popularity of portable devices equipped with two or more communication interfaces. In this new scenario, a key challenge is the choice of network access and technology that best meets the users requirements. Nowadays, the network selection is based on performance criteria, as the strength of the transmission signal. However, with the advance in communication networks intended to provide highly efficient communication to end users, high transmission rates, security and Quality of Experience (QoE), it is necessary to employ criteria that represent more the users needs, in order to assist the access decision. In this direction, this work presents a decision-making system for the access network selection based on Markov Decision Process, in which the current choice does not depend on the historical choices. The system employs both performance and security criteria in the analysis of network quality. The Network Simulator 3 (NS3) has been used in the system evaluation, and the analysis of the results shows a reduction in the number of unnecessary transitions and effectiveness in choosing the most reliable network.*

Resumo. *A sobreposição de diversas tecnologias de comunicação sem fio, como Wi-Fi e LTE, em uma mesma região geográfica tem aumentado com a popularização de dispositivos portáteis dotados de duas ou mais interfaces de comunicação. Diante desse novo cenário, um dos principais desafios consiste na escolha da rede de acesso que melhor satisfaça os requisitos dos usuários. Atualmente, a seleção da rede ocorre apenas com base em critérios de desempenho, como a força do sinal de transmissão. No entanto, com os avanços das redes, visando fornecer aos usuários alta eficiência na comunicação, altas taxas de transmissão, segurança e Qualidade de Experiência (QoE), faz se necessário o emprego de critérios com características mais representativas das necessidades dos usuários para auxiliar na decisão de acesso. Contribuindo nesta direção, este trabalho apresenta um sistema de decisão para seleção da rede de acesso fundamentado no Processo de Decisão Markoviano, em que a escolha atual não depende do histórico das escolhas. O sistema emprega simultaneamente critérios de desempenho e de segurança na análise da qualidade das redes. O simulador NS3 foi utilizado na avaliação do sistema, e a análise dos resultados mostra uma redução no número de transições desnecessárias, além da eficácia na escolha da rede mais confiável.*

1. Introdução

Diante da necessidade de prover mobilidade com suporte à conectividade contínua para dispositivos em trânsito, as redes de acesso sem fio passaram por uma evolução expressiva

nas últimas décadas [Louta and Bellavista 2013]. No passado, as redes de acesso sem fio eram reduzidas, e a manutenção da conectividade se restringia a área de cobertura de cada tecnologia de comunicação utilizada pela rede. Atualmente, a proliferação dessas redes fez aumentar as áreas de transmissão sobrepostas, principalmente em razão da expansão das redes heterogêneas sem fio, dificultando a seleção da rede para o estabelecimento e manutenção da conexão contínua.

As redes heterogêneas sem fio possibilitam que usuários portadores de dispositivos computacionais móveis, habilitados com múltiplas interfaces de comunicação, transitem por diferentes áreas de transmissão e estejam sempre conectados por algum tipo de rede de acesso, tais como Wi-Fi, GSM, LTE. Contudo, um dos principais desafios desse novo cenário corresponde à escolha da rede para o estabelecimento e manutenção da conexão. Apesar da necessidade de mobilidade e manutenção da conectividade ainda ser relevante, outras exigências passam a fazer parte da demanda dos usuários. Atributos de segurança como confidencialidade, integridade e disponibilidade tornam-se cada vez mais importantes para a escolha da rede de acesso. A seleção da rede mais confiável entre as redes disponíveis se torna fundamental para a garantia da qualidade dos serviços utilizados, segurança das transações realizadas na rede e a satisfação dos usuários.

Na literatura, inúmeros trabalhos discutem vários métodos e técnicas de decisão e escolha da rede de acesso em uma ambiente de redes heterogêneas sem fio sobrepostas [Ahmed et al. 2014, Márquez-Barja et al. 2011, Kassar et al. 2008, Martínez-Morales et al. 2010, Yan et al. 2010]. Entre esses métodos de decisão destacam-se os monocritério e multicritério. Os métodos de decisão monocritério consideram apenas um parâmetro no processo de seleção da rede. A força do sinal recebido RSS *(Received Signal Strength)* consiste no critério mais utilizado para a avaliação da qualidade das redes. Já os métodos multicritério MCDM *(Multiple-Criteria Decision-Making)* consideram diferentes parâmetros para a tomada de decisão de seleção da rede de acesso. No entanto, critérios de segurança não são explorados, e quando o são, as abordagens apresentam apenas um peso de importância para o critério e não detalham um método de avaliação das condições de segurança da rede [Melo et al. 2014]. Em geral, a seleção da rede de acesso sem fio considera critérios de desempenho associados a qualidade do sinal de transmissão e não as condições de segurança das redes.

A dinamicidade das redes heterogêneas sem fio dificulta uma avaliação precisa sobre as reais condições de desempenho e segurança da rede. Uma avaliação a partir do histórico das redes pode não representar a situação atual e ocultar informações relevantes para uma decisão confiável. O *Processo de Decisão de Markov MDP (Markov Decision Process)* compreende uma técnica que utiliza informações momentâneas para à tomada de decisão. Esta abordagem se adapta a seleção automática da rede em ambientes dinâmicos, oferecendo maior precisão na avaliação e possibilitando uma escolha confiável.

Esse trabalho apresenta um sistema de decisão de acesso (SDA) para seleção de conexão em ambientes de redes heterogêneas sem fio sobrepostas. O sistema utiliza um método multicritério e a escolha da rede segue o *processo de decisão markoviano*. O sistema proposto utiliza como entrada dados coletados da rede indicando suas condições atuais de desempenho e de segurança. Em seguida, um módulo de inteligência cognitiva composto por uma fase de análise e inferência calcula os valores de qualidade de cada rede detectada pelo dispositivo em transição. Por fim, um módulo de seleção escolhe a

rede com base em políticas definidas pelo usuário, rede ou aplicação.

A avaliação do sistema foi realizada por meio de simulações utilizando o *Network Simulator 3* (NS3). Foram utilizados cenários com redes heterogêneas de tecnologia de comunicação Wi-Fi e LTE, com diferentes áreas de sobreposição e diferentes velocidades de mobilidade dos dispositivos em trânsito de uma rede para outra. Os resultados mostram que o SDA reduziu o número de transições desnecessárias dos dispositivos em redes sobrepostas. Também observou-se que o processo de decisão não causa impacto no fluxo de dados dos dispositivos móveis em trânsito e garante a seleção da rede de acesso mais confiável, com base em critérios de segurança, como fragilidade e robustez.

O restante do artigo está organizado da seguinte maneira. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha o sistema proposto e o seu funcionamento. A Seção 4 apresenta uma avaliação de desempenho e a Seção 5 conclui o trabalho.

2. Trabalhos relacionados

Na literatura, inúmeros trabalhos têm apresentado diferentes métodos utilizados no processo de decisão de transição por redes heterogêneas sem fio. Essas abordagem procuram satisfazer o princípio "Sempre Melhor Conectado"(*ABC - Always Best Connected)* [Louta and Bellavista 2013, Gustafsson and Jonsson 2003]. Esses trabalhos podem ser classificados segundo: *i)* o(s) critério(s) utilizado(s) para comparação entre as redes avaliadas e *ii)* os métodos de inferência usados para processar os valores dos critérios considerados [Cheelu et al. 2013, Kassar et al. 2008, Yan et al. 2010, Charilas and Panagopoulous 2010]. Entre os principais critérios utilizados destacam-se: *Força de Sinal Recebido (RSS - Received Signal Strength), Largura de Banda Disponível, Custo Monetário do Acesso, Taxa de Transferência da Rede*, entre outros. Já em relação aos métodos destacam-se as estratégias baseadas em *Função de Custo, Lógica Fuzzy, Redes Neurais Artificiais, Teoria dos jogos e MCDA (Multiple-Criteria Decision Analysis)*.

Marques apresenta um estudo extensivo sobre algoritmos, protocolos e ferramentas utilizadas no processo de transição em redes heterogêneas sem fio[Márquez-Barja et al. 2011]. Ele discute inúmeras propostas para a fase de decisão e apresenta uma taxonomia sobre as abordagens mais utilizadas. Uma visão geral e uma categorização dos diversos esquemas de decisão de transição em redes heterogêneas também foram discutidos em [Cheelu et al. 2013], onde categorizam os diferentes critérios usados no processo de decisão. Nilakshee também apresenta uma classificação das estratégias de decisão a partir da perspectiva dos algoritmos utilizados no processo de seleção [Nilakshee et al. 2013]. Já a classificação proposta por Ahmed [Ahmed et al. 2014] considera cinco classes de esquemas utilizados no processo de seleção da rede de acesso: esquemas com base em *RSS, QoS, função de decisão, inteligência artificial* e esquemas com base em *contexto*. Para cada classe apresentada, inúmeras técnicas e métodos foram discutidos.

Um ponto em comum verificado em todos esses trabalhos sobre a classificação das estratégias de decisão de transição para redes heterogêneas foi a ausência da utilização de critérios de segurança na avaliação das redes. A segurança não foi considerada com a devida importância. Todos os trabalhos citados não discutem questões relacionadas à segurança. Alguns raros trabalhos [Márquez-Barja et al. 2011, Ahmed et al. 2014] alertam para a importância do uso da segurança no processo de decisão e de escolha da melhor rede de acesso. Neste contexto, o sistema de decisão de acesso proposto neste trabalho

considera a segurança como principal fator de decisão. Assim, os critérios relacionados à segurança são priorizados e utilizados juntamente com os critérios tradicionais de desempenho, para garantir uma conexão segura e confiável em redes de acesso heterogêneas sem fio sobrepostas.

3. Sistema de decisão de acesso em redes heterogêneas

As rede heterogêneas, caracterizadas como um conjunto de redes de diferentes tecnologias e padrões de comunicação [Mesquita Souza et al. 2014], possibilitam o oferecimento de conectividade ubíqua aos dispositivos móveis suportadas por transições (*handoff*) transparentes entre uma rede e outra. No entanto, a escolha da melhor rede entre um conjunto de redes disponíveis se torna fundamental para a garantia da qualidade dos serviços utilizados e para a satisfação dos usuários. A escolha equivocada da rede de acesso pode comprometer os serviços, a comunicação e a conectividade dos dispositivos. A segurança dessas redes representa um fator crítico que pode comprometer os dados e as transações dos usuários. Decidir pela conexão na rede que melhor satisfaça os requisitos dos usuários torna-se uma tarefa importante para a manutenção da comunicação fim-a-fim.

3.1. Modelagem do processo de decisão de acesso

A tomada de decisão de acesso em redes heterogêneas sem fio é modelada por um Processo de Decisão de Markov - MDP *(Markov Decision Process)*. Esta técnica permite a representação de processos de transição de estados probabilísticos, com a possibilidade de observação do estado e possível interferência, executando ações em épocas de decisão [Puterman 1994]. Neste método, cada ação possui uma recompensa ou custo, dependendo do estado em que o processo esteja. O MDP obedece a propriedade de Markov, em que o efeito de uma ação em um estado depende apenas da ação e do estado atual, sem necessidade de informações de ações e estados anteriores. A técnica é conhecida como um *"processo de decisão"* pois modela a possibilidade de um agente tomador de decisões interferir periodicamente no sistema executando ações, diferentemente das *Cadeias de Markov*, onde não há possibilidade de interferência no processo [Puterman 1994].

A existência de diferentes tipos de redes por onde o usuário transita representa o *ambiente de decisão*. A quantidade de redes e suas respectivas qualidades são representados por *eventos aleatórios*, que variam de acordo com o tempo. As *ações* que podem ser executadas pelo processo de decisão consistem de *monitoração, conexão e desconexão* da rede. O *estado* em que o dispositivo móvel pode estar consiste de *conectado* ou *desconectado*. A *política de decisão* corresponde a conexões em redes mais seguras e confiáveis durante o percurso do usuário. A satisfação do usuário consiste na *recompensa* pela conectividade contínua e segura. Essa definição do problema juntamente com a extração de características bem definidas das redes permitem a sua representação por meio do MDP. Formalmente o MDP consiste de uma *4-tupla (S, A, T, R)* onde:

- S = representa o conjunto de estados em que o processo (sistema) pode estar;
- A = compreende o conjunto de ações que podem ser executadas em diferentes épocas de decisões;
- $T : S \times A \times S \longmapsto [0, 1]$ implica em uma função de probabilidade de um sistema passar para o estado $s' \in S$, dado que o processo esteja em um estado $s \in S$ e o agente tomador de decisão decidiu executar uma ação $a \in A$ (denotada por $T(s' \mid s, a)$);

- $R : S \times A \longmapsto \mathbb{R}$ representa uma outra função do custo ou recompensa por tomar uma decisão $a \in A$ quando o processo estiver em um estado $s \in S$.

A cada *época de decisão (d)*, o agente tomador de decisões usa uma *regra de decisão (r)* para escolher a próxima *ação (a)*. Uma forma simples de regra de decisão consiste em um mapeamento direto de estados em ações como $r : S \longmapsto A$. Assim, uma regra de decisão r para um MDP em uma época de decisão d consiste de uma função $rd : S \longmapsto A$, que determina a ação a ser executada, dado o estado do sistema. O conjunto de todas as regras de decisão (uma para cada época de decisão) é chamado de *Política*. Normalmente o principal objetivo do processo consiste em encontrar uma política que otimize um dado critério de desempenho [Pellegrini and Wainer 2007]. Uma política pode ser classificada como: Markoviana (ou sem Memória), quando a escolha da ação depende apenas do estado corrente; e Não-Markoviana, quando a escolha da ação depende de todo o histórico de ações e estados do sistema até o momento. O processo de decisão de acesso em redes heterogêneas é ilustrado na Figura 1. Neste trabalho, a melhor rede de acesso representa a rede mais segura e confiável.

- S= [Conectado, Desconectado]
- A= [Monitoração, Conexão, Desconexão]
- $T : S \times A \times S \longmapsto [0, 1]$ probabilidade de estar em um estado de S
- $R : S \times A \longmapsto \mathbb{R}$ custo ou recompensa por tomar uma decisão
- $\pi-$PoliticaOptimal = melhor rede

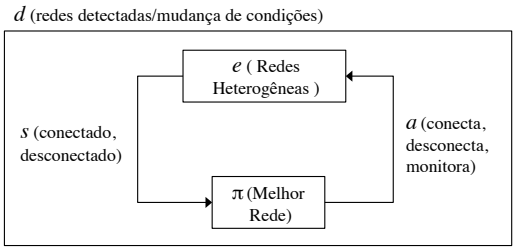


Figura 1. Dinâmica de funcionamento de um MDP em redes heterogêneas

O agente tomador de decisões consulta o ambiente e (redes heterogêneas), verifica o estado atual do sistema s (conectado ou desconectado), consulta uma política optimal π (conexão em melhor rede entre redes disponíveis) e executa uma ação a (monitora novas redes, conecta ou desconecta). Esse ciclo se repete em todas as épocas de decisões. O ciclo operacional do agente tomador de decisão pode ser encadeado e contínuo nas diferentes épocas de decisões. Esse processo cíclico não necessita do conhecimento dos estados anteriores do sistema e utiliza apenas informações atuais, obedecendo o princípio markoviano. A Figura 2 ilustra o funcionamento do processo MDP de ciclo contínuo.

Em uma época de decisão (d), o ambiente (e) possibilita ao agente tomador de decisão verificar um estado (s_j) do sistema que leva a uma política (π). A partir da política (π) uma ação (a_j) pode induzir o sistema a um novo ambiente, ou a um ambiente anterior, executando a ação (a_i). Neste novo ambiente (e), um estado (s_j) leva o sistema a uma nova política, ou a uma política anterior, com o estado (s_i). Esse processo se repete durante todas as épocas de decisões (d), assegurando a propriedade markoviana, de modo que uma decisão não depende de informações de ações nem de estados anteriores, bastando apenas conhecer a situação atual do sistema para a tomada de decisão.

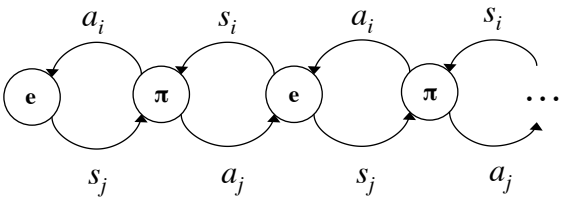


Figura 2. Dinâmica de funcionamento de um MDP de ciclo contínuo

3.2. Arquitetura do sistema de decisão de acesso

O sistema de tomada de decisão tem como objetivo auxiliar os usuários de dispositivos computacionais móveis a escolher e transitar por redes de acesso heterogêneas com conectividade segura e contínua. Para isso, o sistema utiliza dados das condições atuais das diferentes redes detectadas e realiza um processo de inferência para decidir qual a rede que melhor satisfaz seus requisitos entre todas as redes disponíveis. Diferente das estratégias existentes na literatura, as quais utilizam apenas critérios relacionados ao desempenho, nesta abordagem a escolha da rede considera os critérios relacionados à *conectividade* e *segurança* simultaneamente como principais fatores de decisão.

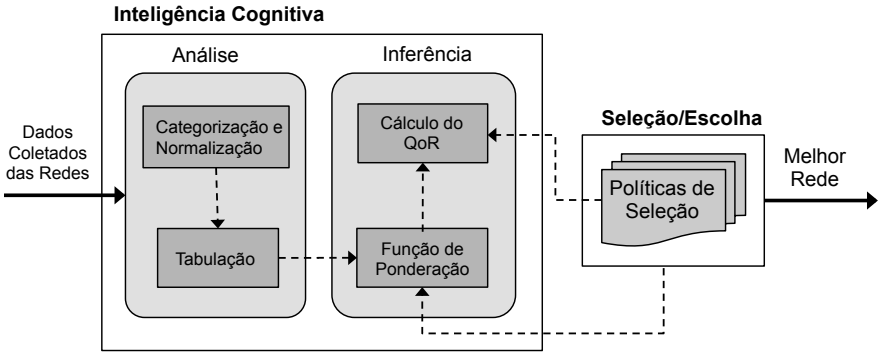


Figura 3. Arquitetura do sistema de decisão de acesso

O sistema de tomada de decisão de acesso em redes heterogêneas sem fio compreende dois módulos, como ilustrado na Figura 3. O primeiro denominado de *Inteligência Cognitiva*, tem a função de processar os dados coletados do ambiente das diferentes redes detectadas, para inferir a melhor rede disponível, sob o aspecto de segurança e conectividade. A conectividade é avaliada por meio de critérios tradicionais de desempenho com RSS. Já a segurança utiliza indicadores apoiados na *fragilidade* e na *robustez* da rede. Esses indicadores foram detalhados em [Melo et al. 2014]. O segundo módulo da arquitetura, *Seleção/Escolha*, seleciona a rede mais segura e confiável durante a transição dos usuários móveis pelas redes de tecnologias de comunicação heterogêneas.

O módulo de *Inteligência Cognitiva* compreende os componentes de *Análise e Inferência*. O componente de *Análise* corresponde à primeira fase para o cálculo da qualidade rede. Esse componente envolve os elementos de *Categorização e Normalização, e a Tabulação*. O elemento *Categorização e Normalização* separa os dados coletados em diferentes categoria e aplica técnicas de padronização dos valores a fim normalizá-los em mesma ordem de medida. O elemento *Tabulação* recebe os dados normalizados e os or-

ganiza em um conjunto que será encaminhado para a fase de *Inferência* e utilizado para o cálculo da qualidade da rede.

O elemento *Categorização e Normalização* utiliza a razão entre o valor atual de um determinado critério (c) e o valor ótimo (nominal) desse mesmo critério referente à rede analisada. A fórmula para esse processo é definida pela Equação 1.

$$NCi = \frac{ValorAtual}{ValorOtimo} \quad \forall \quad i = \{1, 2, 3, ..., c\} \tag{1}$$

Onde, i implica no conjunto de critérios utilizados para a análise da rede, e NCi corresponde à normalização do *i-ésimo* critério. A normalização feita por NCi é dada pela razão entre o *valor atual* de um critério (c), obtido no momento de funcionamento da rede, e o *valor ótimo* do critério para determinada tecnologia da rede. Note que os valores normalizados para cada critério estão no intervalo entre 0 e 1.

O elemento de *Tabulação* realiza uma operação de organização dos valores normalizados em um conjunto de critérios Cc. Após serem organizados esses valores são encaminhados para uma função de ponderação, onde receberão pesos de importância. O elemento de *Tabulação* obedece a Equação 2, onde Cc corresponde ao conjunto de critérios já normalizados, utilizados para a análise das condições da rede.

$$\begin{aligned} Cc &= [0]; \\ Cc &= Cc \bigcup NCi \quad \forall\, 1 \leq i \leq c \end{aligned} \tag{2}$$

A *Inferência* executa a segunda fase do processo de decisão. Esse componente é responsável pelo cálculo do índice de qualidade da rede. O componente é formado pelos elementos *Função de Ponderação e Cálculo da Qualidade da Rede (QoR)*. A Função de *Ponderação* atribui pesos de importância para os critérios utilizados na avaliação da rede. Neste trabalho cada peso w da *Função de Ponderação* é representado em um intervalo [0,1] e a soma dos pesos deve ser igual a 1, como mostra a Equação 3.

$$\begin{aligned} \sum_{j=1}^{k} w_i = 1 \; onde \; w_i > 0 \; \forall \; i \\ Cp = [w_1, w_2, w_3, ...w_n] \\ |Cp| = |Cc| \\ \forall \; i \in Cp \; \exists \; j \in Cc \end{aligned} \tag{3}$$

Onde, Cp consiste no conjunto de pesos de importância representados por $w_1, w_2, w_3, ..., w_n$. Cc corresponde ao conjunto de critérios. A Cardinalidade de Cp é igual a de Cc. O somatório dos elementos de Cp deve ser igual a 1, e para cada elemento de Cp existe um elemento correspondente em Cc.

O *Cálculo do QoR* pode utilizar diferentes técnicas mono ou multicritério para inferir um índice de qualidade da rede com base nos critérios e métricas utilizadas na

avaliação. Neste trabalho, o processo de inferência de melhor rede utiliza uma abordagem multicritério inspirada no método NWAUF *(Normalized Weighted Additive Utility Function)* [Malakooti and Thomas 2006], fundamentado nos estudos que mostram o seu baixo custo computacional em comparação a outros métodos MCDA *(Multiple-criteria decision analysis)*, como o AHP *(Analytic Hierarchy Process)* [SAATY 1980] ou ELECTRE *(ELimination and Choice Expressing REality)* [Vahdani et al. 2010], e as suas respectivas variantes [Malakooti and Thomas 2006, Wang and Triantaphyllou 2008]. O método NWAUF utiliza múltiplos critérios normalizados e com pesos de importância para calcular uma função de utilidade. Essa função considera o somatório dos pesos utilizado juntamente com os critérios normalizados. A função de utilidade calculada determina o valor da qualidade da rede. A Equação 4 ilustra a fórmula para o *Cálculo do QoR*. Onde, QoR corresponde à qualidade da rede verificada pela soma de cada critério NCi multiplicado por seu peso de importância w_i na avaliação das condições da rede.

$$QoR = \sum_{i=1}^{c} w_i \times NCi, \ \forall \, i = 1, 2, 3, ..., c \tag{4}$$

O segundo módulo do sistema de tomada de decisão de acesso corresponde à *Seleção e Escolha*. Este módulo tem como base um componente de *Políticas de Seleção*. Essas políticas definem qual rede será selecionada para o acesso do dispositivo em transição. Caso não exista nenhuma rede que satisfaça plenamente todas as características da política definida, a seleção se dará pela rede que possuir condições mais próximas das desejadas, garantindo assim a conectividade ao dispositivo móvel.

O componente *Políticas de Seleção* aponta a escolha de determinada rede, assim como, as redes podem definir a alteração de uma política de seleção devido às condições do ambiente. Neste trabalho, a política de seleção de acesso é fortemente influenciada pelas condições de conectividade e segurança da rede. Logo, a seleção de acesso opta pelas redes mais seguras e conexas. Vale ressaltar que uma determinada política de seleção pode influenciar a atribuição dos pesos de importância da *Função de Ponderação* do componente de *Inferência*. Esse processo permite uma seleção da rede de modo mais personalizado e de acordo com características e preferências dos usuários e dos ambientes heterogêneos.

3.3. Funcionamento do sistema de decisão de acesso

A demonstração do funcionamento do sistema de decisão de acesso considera um ambiente de redes heterogêneas sem fio sobrepostas e usuários portadores de dispositivos móveis em trânsito por estas redes. Ao se encontrar em uma área sobreposta por duas ou mais redes, o sistema deve decidir qual rede de acesso selecionar e estabelecer a conexão. Para isso, utiliza informações das condições momentâneas de cada rede. Este processo se repete em todo instante de detecção de uma nova rede em sua área de cobertura. A Figura 4 ilustra um cenário de funcionamento do sistema de decisão de acesso.

O sistema de decisão de acesso tem como entrada dados das condições momentâneas das redes coletados pelo próprio dispositivo móvel ou disponibilizados pelos provedores de acesso da rede detectada. Neste exemplo, os dados da rede utilizados para a avaliação são *Tipo de Tecnologia de comunicação, Endereço Lógico, Endereço Físico,*

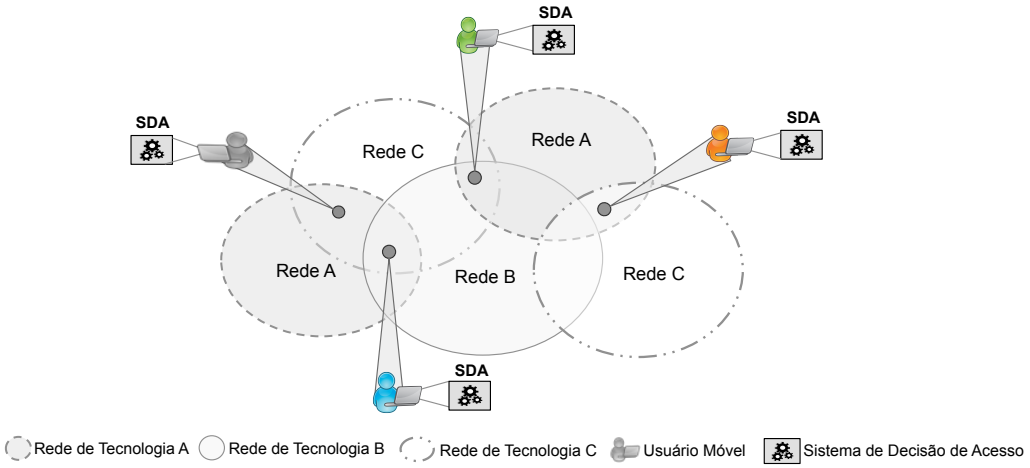


Figura 4. Cenário de funcionamento do sistema de decisão de acesso

Taxa de Entrega momentânea, Força do Sinal Recebido - RSS (Received Signal Strength) e um Indicador de Segurança da rede [Melo et al. 2014]. A partir desses dados, o componente de *Análise* os organiza em uma lista de critérios normalizados em mesma unidade de medida. A normalização ocorre por meio da fração entre o valor momentâneo e o valor nominal de cada critério observado em cada tecnologia de rede. Neste momento, uma função de ponderação recebe os valores desta lista e atribui pesos de importância para cada um dos critérios considerados na avaliação. No exemplo de funcionamento do sistema de decisão de acesso os pesos foram distribuídos da seguinte forma: $\alpha = 0,25$, $\beta = 0,25$ e $\delta = 0,50$. Esses pesos de importância foram definidos de modo a garantir uma decisão com prioridade na segurança, cujo peso se refere a δ.

O resultado da função de ponderação é encaminhado para o componente de *Inferência*, que calcula a qualidade da rede (QoR) com base em uma função de utilidade obtida pela soma ponderada dos critérios de avaliação com seus respectivos pesos de importância. O resultado do QoR de cada rede é armazenado em uma lista de redes. Por fim, o componente de *Seleção* decide pela conexão na rede de acesso que possuir o maior valor para o QoR. A Figura 5 ilustra todo o fluxo de operações realizadas no funcionamento do sistema de decisão de acesso.

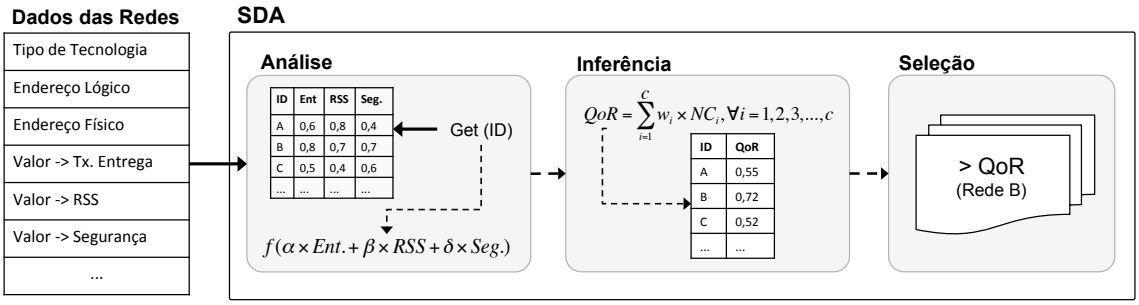


Figura 5. Fluxo de operações do sistema de decisão de acesso

4. Avaliação de desempenho do sistema de decisão de acesso

O sistema foi implementando no simulador NS-3.21 e as redes simuladas empregam tecnologias de comunicação Wi-Fi e LTE. Essas redes heterogêneas são representativas para a execução de transições de conexões. A avaliação do sistema de decisão de acesso considera três cenários distintos com diferentes velocidades de mobilidade dos dispositivos, e diferentes áreas de sobreposição de redes heterogêneas sem fio. A variação da velocidade do usuário corresponde a um *pedestre (1m/s)*, um *ciclista (3m/s)* e um *motorista de automóvel (12m/s)*. A área de sobreposição das redes utilizada compreende um ambiente *doméstico* como uma casa que possui *baixa sobreposição de redes* BSR (duas redes sobrepostas), uma área *residencial* como um bairro ou um conjunto de apartamentos em um prédio, com *média sobreposição de redes* MSR (dez redes sobrepostas) e uma região de um *centro comercial* com *alta sobreposição de redes* ASR (vinte redes sobreposta), como ilustrado na Figura 6. A Tabela 1 apresenta os parâmetros utilizados em cada cenário.

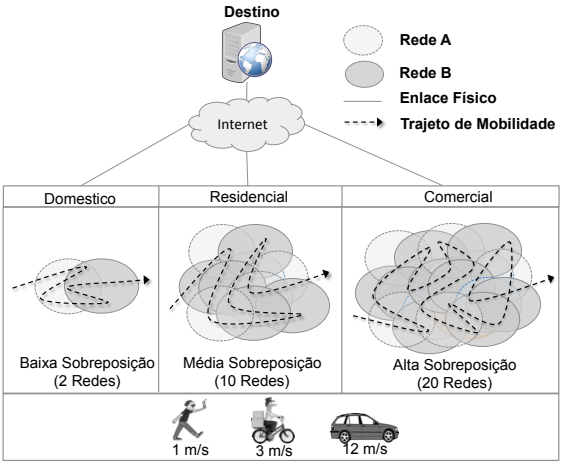


Figura 6. Cenários de simulações

Parâmetros	Valores	
Áreas de Cobertura	Domestico	100m x 100m
	Residencial	150m x 150m
	Comercial	200m x 200m
Sobreposição de Redes	$Baixa$	2
	$Media$	10
	$Alta$	20
Raio de Cobertura	100m	
Interfaces de Comunicação	Wi-Fi e LTE	
Velocidade de Mobilidade	1m/s; 3m/s; 12m/s	
Largura de Banda	10 Mbp/s	
Aplicação	CBR sobre UDP	

Tabela 1. Parâmetros de simulação

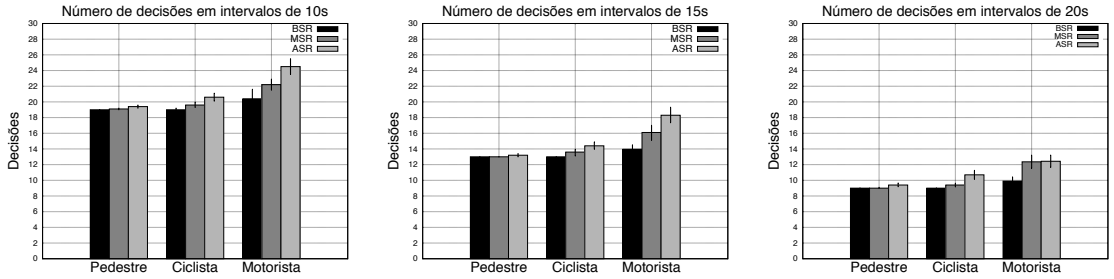
As métricas utilizadas na avaliação foram: *Número de decisões* - indica a quantidade de decisões de acesso realizada durante o trânsito do dispositivo pelas redes heterogêneas; *Número de transições* - mostra o número de transições realizadas a partir das decisões de escolha da rede; *Tempo de decisão* - verifica o tempo gasto para decidir pela melhor rede de acesso disponível. Os resultados foram obtidos a partir da média de 30 rodadas de simulações com intervalos de confiança de 95%.

4.1. Número de decisões

O número de decisões sobre as redes de acesso foi verificado considerando intervalos de 10, 15 e 20 segundos. Esses intervalos são utilizados para consulta das condições das redes, que devido à dinamicidade podem ser alteradas a todo instante. Assim, as decisões de seleção da rede acontecem a medida que eventos, como a detecção de novas redes, mudanças das condições da rede atual, a saída da área de cobertura de uma rede ou até que os valores dos intervalos sejam obedecidos.

A Figura 7 ilustra os gráficos do número de decisão para cada intervalo, variando a velocidade do dispositivo móvel nos três cenários de avaliação: *BSR, MSR e ASR*. O gráfico da Figura 7(a) mostra que, com a velocidade de um *pedestre*, o número de decisões pelas redes de melhor qualidade se manteve constante independente do tipo de cenário

avaliado. Já com a velocidade de um *ciclista*, existe apenas um aumento do número de decisões quando o número de sobreposições de redes é alto. Na velocidade de um *motorista*, foi verificada uma variação do número de decisões de acordo com cada cenário avaliado. Os resultados permitem observar que quando a velocidade do dispositivo móvel aumenta e o número de sobreposições de redes também, o número de decisões realizadas pelo SDA é incrementado. Isso ocorre porque o dispositivo em transição detecta mais redes durante sua mobilidade e precisa escolher mais vezes a rede de acesso.



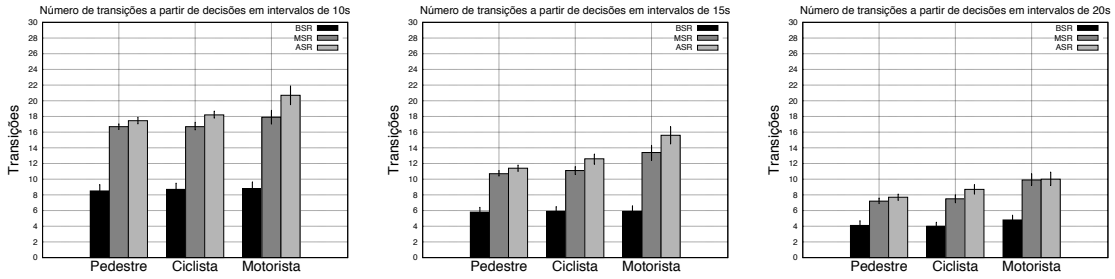
(a) (b) (c)

Figura 7. Avaliação do número de decisões

O mesmo comportamento do gráfico da Figura 7(a) também é encontrado nas Figuras 7(b) e 7(c). Contudo, os valores relacionados ao número de decisões são inferiores devido ao aumento do intervalo de decisões do sistema. Na Figura 7(c), com um intervalo de 20s, o dispositivo móvel com velocidade de um *motorista* estabiliza o número de decisões nos cenários *MSR e ASR*. Isso ocorre porque mesmo que a sobreposição e a dinamicidade das redes sejam média e alta, o intervalo de decisão de 20s faz com que as consultas sobre as condições das redes sejam menores, e consequentemente o número de decisões também.

4.2. Número de transições

O número de transições de acesso de uma rede para outra foi avaliado considerando o mesmo ambiente utilizado para a avaliação do número de decisões. Nesta avaliação, as transições de acesso só ocorrem a partir da decisão da escolha da rede mais segura entre as redes disponíveis. A transição de uma rede para outra implica na manutenção da conectividade contínua. A Figura 8 ilustra os gráficos do número de transições realizadas a partir das decisões de acesso nos três cenários de avaliação. O gráfico da Figura 8(a) mostra que independente da velocidade do dispositivo, o número de transições se manteve constante quando a sobreposição de redes é baixa. Em casos de média sobreposição de redes, o número de transição tem uma pequena variação comparando todas as velocidades de mobilidade. Com alta sobreposição, os números de transição são maiores quando as velocidades também são altas, o que pode ser explicado pela constante detecção de novas redes em razão da mobilidade. O gráfico da Figura 8(b) apresenta comportamento similar, mas com valores de transição reduzido em razão do maior intervalo de decisão. O gráfico da Figura 8(c) mostra uma certa estabilidade no número de transições em cenários *MSR*. Os valores de transição entre 10 e 20 redes sobrepostas são similares na maioria dos casos.

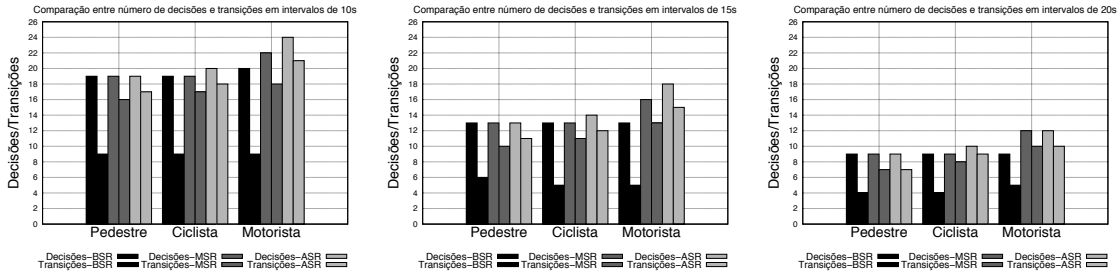


(a) (b) (c)

Figura 8. Avaliação do número de transições a partir das decisões

Número de decisões × número de transições

Os resultados da avaliação do número de decisões e do número de transições possibilitam observar que o sistema de decisão reduz o número de transições desnecessárias em áreas de redes heterogêneas sobrepostas em até 50%. O gráfico da Figura 9 ilustra a comparação do número de decisões e o número de transições realizadas para todos os cenários de simulação e em todos os intervalos de consulta das condições da rede e decisão de acesso. Em todos os resultados ilustrados nas Figuras 9(a), 9(b) e 9(c) o número de transições é inferior ao número de decisão, o que indica que decidir pela rede de melhor qualidade pode evitar transições desnecessárias e garantir conectividade contínua e segura.



(a) (b) (c)

Figura 9. Número de decisões e número de transições realizadas

4.3. Tempo de decisão

A avaliação do tempo de decisão mostra a eficiência do sistema em decidir pela rede de acesso mais segura nos cenários utilizados na simulação. Em cada cenário foram considerados os intervalos de decisão referentes a 10, 15 e 20 segundos. Os resultados ilustrados no gráfico da Figura 10 mostram que o tempo de decisão é similar em todos os intervalos. Contudo, existe um acréscimo do tempo de decisão quando aumenta o número de redes sobrepostas, fazendo com que o sistema procure analisar mais redes na mesma área de cobertura do dispositivo móvel. Outro ponto destacado pelos resultados consiste no baixo tempo necessário para decidir qual é a melhor rede disponível. No pior caso, em que existe alta sobreposição de redes o tempo gasto para decidir sobre a melhor rede corresponde a 270 milissegundos. Já no melhor caso, com apenas duas redes sobrepostas o tempo de decisão é de 150 milissegundos. Esses resultados mostram que o sistema de

tomada de decisão não tem impacto significativo na sobrecarga de tempo de transição de uma rede para outra, independente de sua tecnologia de comunicação.

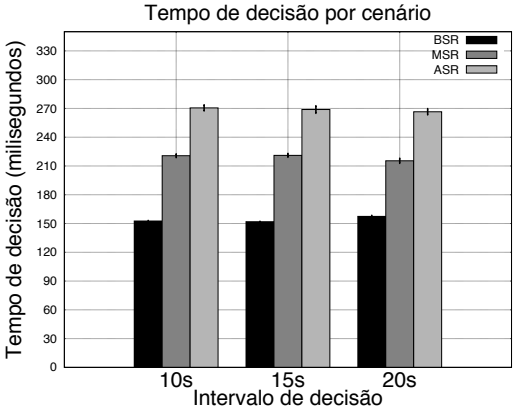


Figura 10. Avaliação do tempo decisão

5. Conclusão

Este trabalho apresentou um sistema de decisão de acesso em redes heterogêneas sem fio sobrepostas. O sistema de decisão possibilita a escolha da rede melhor qualificada, com base em critérios de desempenho e segurança para garantir uma transição de conexão contínua e segura, durante a mobilidade por diferentes redes de acesso. O sistema utiliza o processo de decisão markoviano, em que condições prévias não são relevantes para a decisão de conexão. Está estratégia foi utilizada para atender a dinamicidade das redes heterogêneas, que são constantemente alteadas pelas associações e dissociações de dispositivos em trânsito. A avaliação foi realizada por simulações no NS3 utilizando redes heterogêneas com tecnologias de comunicação Wi-Fi e LTE. Os cenários considerados possuem diferentes velocidades de mobilidades dos dispositivos e diversificadas áreas de sobreposição de redes para permitir uma avaliação do processo de decisão. Os resultados obtidos através das métricas *número de decisões, número de transições e tempo de decisão* indicam uma melhoria na redução do número de transições desnecessárias e de transição por redes de acesso seguras e confiáveis, sem a necessidade de uma sobrecarga no tempo de decisão. Em trabalhos futuros pretende-se explorar novos métodos de avaliação multicritério para auxiliar na tomada de decisão e contribuir para a manutenção de conectividade de dispositivos em trânsito por redes heterogêneas seguras e confiáveis.

Referências

Ahmed, A., Boulahia, L., and Gaïti, D. (2014). Enabling vertical handover decisions in heterogeneous wireless networks: A state-of-the-art and a classification. *IEEE Communications Surveys Tutorials*, 16(2):776–811.

Charilas, D. and Panagopoulous, A. (2010). Multiaccess radio network enviroments. *IEEE Vehicular Technology Magazine*, 5(4):40–49.

Cheelu, D., Babu, M. R., and Krishna, P. V. (2013). Study of vertical handoff decision strategies in heterogeneous wireless networks. *International Journal of Engineering & Technology*, 5(3).

Gustafsson, E. and Jonsson, A. (2003). Always best connected. *IEEE Wireless Communications*, 10(1):49–55.

Kassar, M., Kervella, B., and Pujolle, G. (2008). An overview of vertical handover decision strategies in heterogeneous wireless. *Computer Communications*, 31(10):2607 – 2620.

Louta, M. and Bellavista, P. (2013). Bringing always best connectivity vision a step closer: challenges and perspectives. *IEEE Communications Magazine*, 51(2):158–166.

Malakooti, B. and Thomas, I. (2006). A distributed composite multiple criteria routing using distance vector. In *IEEE International Conference on Networking, Sensing and Control (ICNSC '06)*, páginas 42–47.

Márquez-Barja, J., Calafate, C. T., Cano, J.-C., and Manzoni, P. (2011). An overview of vertical handover techniques: Algorithms, protocols and tools. *Computer Communications*, 34(8):985–997.

Martínez-Morales, J., Pineda-Rico, U., and Stevens-Navarro, E. (2010). Performance comparison between MADM algorithms for vertical handoff in 4G networks. In *IEEE International Conference on Electrical Engineering Computing Science and Automatic Control*, páginas 309–314.

Melo, R., Nogueira, M., and Santos, A. (2014). Sistema indicador de resiliência na conectividade de redes heterogêneas sem fio. In *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (XV SBSeg)*, páginas 139–152.

Mesquita Souza, B., Guidoni, D., Hojo de Souza, F., and Robson Mateus, G. (2014). Topological design of resilient heterogeneous networks with qos. In *Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, páginas 183–190.

Nilakshee, R., Bhavna, A., and Dhande, A. P. (2013). Survey of vertical handover decision algorithms. *International Journal of Innovations in Engineering and Technology (IJIET)*, 2(1):1848–1863.

Pellegrini, J. and Wainer, J. (2007). Processos de decisão de markov: um tutorial. *Revista de Informática Teórica e Alicada (RITA)*, páginas 133–179.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience.

SAATY, T. L. (1980). *The analytic hierarchy process*. McGraw Hill.

Vahdani, B., Jabbari, A., Roshanaei, V., and Zandieh, M. (2010). Extension of the electre method for decision-making problems with interval weights and data. *The International Journal of Advanced Manufacturing Technology*, páginas 793–800.

Wang, X. and Triantaphyllou, E. (2008). Ranking irregularities when evaluating alternatives by using some electre. *Omega*, páginas 45–63.

Yan, X., Sekercioglu, Y. A., and Narayanan, S. (2010). A survey of vertical handover decision algorithms in fourth generation heterogeneous wireless networks. *Computer Networks*, 54(11):1848–1863.

Caracterização da Transmissão de um Grande Evento Esportivo

Wagner Almeida[1,3] **Breno Santos**[1] **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}@ice.ufjf.edu.br {alex.borges}@ufjf.edu.br
{cunha, jussara}@dcc.ufmg.br

Abstract. *The unavailability of multicast in the Internet limits the scalability of streaming popular content like sports events. To alleviate this problem, content providers use alternative approaches like peer-to-peer distribution and content distribution networks. Capacity planning is imperative in both cases and, if done incorrectly, may lead to high costs, performance problems, and frustrated users. In this paper we characterize traffic logs collected from video servers at one of the largest online content providers in South America that streamed FIFA's 2013 Confederations Cup live. We quantify the workload on the streaming servers as well as the quality of service experienced by end users. We show that streaming servers receive substantial load and that load can vary significantly between games, making capacity planning challenging.*

Resumo. *A falta de implantação de multicast na Internet gera problemas de escalabilidade para distribuição de conteúdo popular como transmissão de eventos esportivos. Isto leva à utilização de soluções alternativas como distribuição em redes par-a-par ou por redes de distribuição de conteúdo (CDNs). Em ambos os casos é necessário planejamento de capacidade, que se não for feito de maneira correta pode gerar erros de provisionamento que resultam em problemas de desempenho e usuários frustrados. Neste artigo apresentamos uma caracterização do tráfego da transmissão ao vivo da Copa das Confederações da FIFA de 2013 realizada por um dos maiores provedores de conteúdo multimídia pela Internet da América Latina. Nós quantificamos o impacto da transmissão de um evento de grande escala no provedor de conteúdo e a qualidade de serviço recebida pelos usuários finais. Mostramos também que os servidores de vídeo recebem carga substancial e que a carga varia significativamente entre jogos, o que dificulta o provisionamento de capacidade.*

1. Introdução

Muitos espectadores estão deixando sistemas tradicionais de televisão, como cabo e satélite, e optando por assistir seus programas favoritos sob demanda através de serviços de *streaming* como Netflix [Mahanti 2014]. Neste cenário, sistemas de *streaming* ao vivo vêm recebendo atenção substancial, com um número expressivo de aplicações comerciais emergindo. Entretanto, devido à falta de implantação de *multicast* na Internet, a

distribuição de conteúdo multimídia em larga escala é um desafio significativo. Diversos mecanismos como *streaming* adaptável, uso de redes par-a-par e redes de distribuição de conteúdo (CDNs) vêm sendo utilizados para tornar possível a distribuição de mídia ao vivo em larga escala.

No entanto, ainda são necessários grandes investimentos em infraestrutura—como planejamento de capacidade e provisionamento de recursos—para atender à demanda crescente por esse tipo de conteúdo. Alguns eventos atraem um número muito grande de espectadores, impondo aos sistemas dos provedores de conteúdo e à infraestrutura de rede uma carga de trabalho difícil de prever e que pode exigir mais recursos do que aqueles já provisionados.

Entre as transmissões ao vivo com potencial para atrair grande interesse estão os eventos esportivos. Em particular, os jogos da Copa do Mundo (e seu torneio preparativo, a Copa das Confederações) atraíram a atenção do mundo todo e, devido à paixão local pelo esporte, em especial do Brasil. Espera-se que a transmissão dos jogos olímpicos de 2016 tenha popularidade similar.

Uma tarefa desafiadora no planejamento de capacidade de provedores de conteúdo é prever o volume de tráfego e provisionar banda de rede suficiente para eventos futuros. O volume de tráfego e provisionamento são diferentes para cada provedor, dependendo de questões como localização geográfica, parcerias de rede e número de clientes. As cargas também são diferentes para propriedades específicas do evento como horário e popularidade dos times. O provisionamento de recursos deve ainda considerar requisitos de aplicação, como baixa latência de distribuição.

Neste artigo, apresentamos uma caracterização da transmissão ao vivo de todos os jogos da Copa das Confederações, que é um grande evento preparatório para a Copa do Mundo da FIFA. Nosso trabalho é baseado em *logs* coletados nos servidores de vídeo de um dos maiores e mais conhecidos provedores de conteúdo da América Latina, o Globo.com. Nosso objetivo é entender qual é o impacto de um evento dessa magnitude na carga imposta aos servidores do provedor de conteúdo e na infra-estrutura de rede, bem como avaliar a qualidade de serviço entregue aos usuários.

Nossos resultados mostram que um evento como a Copa das Confederações tem potencial de atrair centenas de milhares de espectadores simultâneos. O tráfego gerado por transmissões de eventos ao vivo impõe desafios de infraestrutura, planejamento e previsão de carga até mesmo para um grande provedor; por exemplo, observamos que o provedor reajustou a qualidade da mídia disponível durante a transmissão de um jogo devido à alta carga. Além disso, pudemos observar que as regiões mais desenvolvidas do Brasil obtiveram maior qualidade na transmissão dos jogos. Isso pode ser relacionado à qualidade no acesso à Internet como um todo nessas regiões.

Nossa primeira contribuição é apresentar uma caracterização profunda do *streaming* e da carga de trabalho à qual o provedor de conteúdo é submetido durante um grande evento. Nessa linha, caracterizamos o número de usuários de cada jogo, a taxa de mídia distribuída e a qualidade da mídia. Nossa segunda contribuição é caracterizar a qualidade de serviço recebida pelos clientes e relacioná-la a dados geográficos e sócio-indicadores. Acreditamos que nossos resultados podem ser úteis para provedores de conteúdo ajudando a direcionar decisões de provisionamento de infra-estrutura e planejamento da capacidade

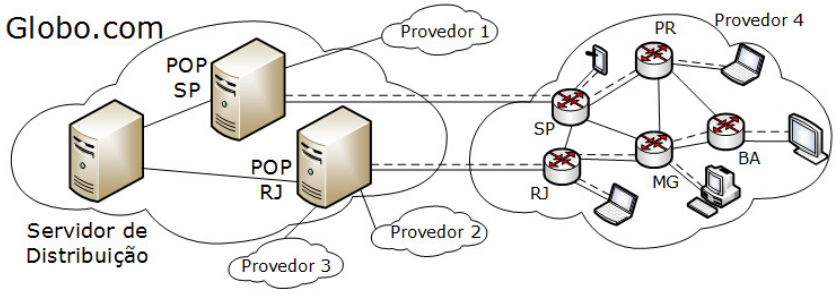


Figura 1. Exemplo de *anycast*

das redes dos provedores de conteúdo visando o *streaming* de grandes eventos.

2. Descrição do Sistema e Conjunto de Dados

Nesta seção analisamos a arquitetura e principais componentes do sistema de transmissão de vídeo ao vivo do Globo.com, bem como o conjunto de dados. A maioria das informações dos *logs* aqui descritas foram fornecidas pelo próprio provedor de conteúdo.

2.1. Infra-estrutura de Distribuição de Conteúdo

O sistema de transmissão de vídeo ao vivo do Globo.com usa *anycast* [Cesario 2012], uma técnica de engenharia de tráfego onde um prefixo IP é anunciado a partir de múltiplos locais. A rede então propaga os anúncios e decide livremente para qual localização encaminhar os dados, considerando a configuração dos protocolos de roteamento e possivelmente a disponibilidade do servidor anunciante [Katabi and Wroclawski 2000].

Como mostramos na figura 1, a infra-estrutura de *streaming* ao vivo possui dois pontos de distribuição em duas cidades pólo do Brasil, Rio de Janeiro e São Paulo. 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. As requisições recebidas em um ponto de distribuição são balanceadas entre os diversos servidores disponíveis em cada um dos pontos de distribuição.

O sistema de transmissão de vídeo do Globo.com já foi utilizado para transmissão de vários eventos de grande escala.Os servidores do Globo.com executam o nginx [1], um servidor Web de alto desempenho.

2.2. Interação Cliente-Servidor

Atualmente, a maioria dos servidores de *streaming* de vídeo, como os servidores do Globo.com, YouTube e Netflix, entregam conteúdo usando HTTP. O uso de HTTP traz consigo a flexibilidade inerente desse ambiente e supera limitações de sistemas de *streaming* em P2P. Por exemplo, transmissão por HTTP requer somente um *browser* padrão para visualizar conteúdos de vídeo, sem necessidade de *software* adicional. Além disso, não é necessário manter portas adicionais TCP ou UDP abertas em *firewalls* e NATs.

Tais sistemas podem disponibilizar múltiplas taxas de codificação com diferentes níveis de qualidade usando *Dynamic Adaptative Streaming over HTTP* (DASH [Stockhammer 2011]). O DASH funciona dividindo o conteúdo de mídia em sequências de segmen-

[1] http://nginx.org/

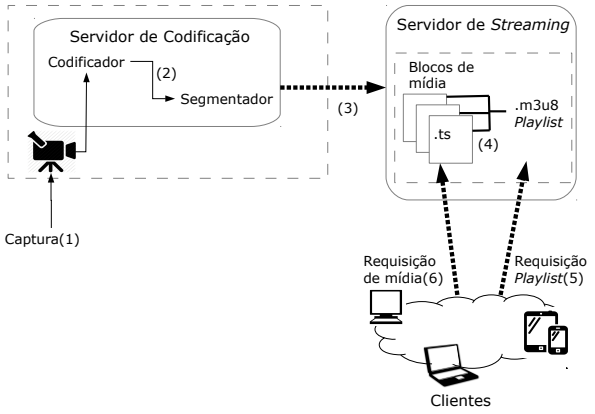


Figura 2. Codificador e segmentador de vídeo

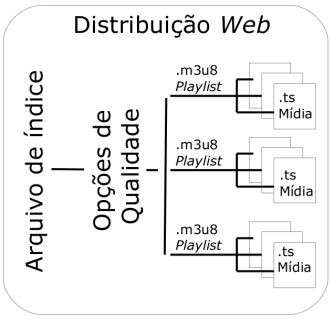


Figura 3. Múltiplas qualidades

tos com taxas de codificação diferentes, de forma que cada cliente pode ajustar continuamente suas requisições de acordo com a estimativa local de disponibilidade de banda.

A transmissão de um evento ao vivo pela Internet não ocorre estritamente em tempo real. É possível observar na figura 2 que é necessário um pré-processamento envolvendo a codificação dos dados de som e imagem. A partir dessa codificação, o DASH divide o arquivo de mídia em múltiplos segmentos de diversas qualidades e gera arquivos de índice em formato *.m3u8*, também chamados de listas de reprodução (figura 3).

Para receber o conteúdo, clientes primeiro solicitam a lista de reprodução referente a ele e então solicitam segmentos da mídia. Cada vez que um segmento é recebido, o cliente pode acessar o servidor e fazer uma nova requisição.

2.3. Conjunto de Dados

Nosso conjunto de dados é composto de arquivos de *log* dos servidores nginx do Globo.com coletados durante transmissão ao vivo de jogos da Copa das Confederações 2013.

A Copa das Confederações é um torneio de futebol preparatório para a Copa do Mundo e foi realizado no Brasil entre 15 e 30 de Junho de 2013. Oito seleções estiveram presentes no torneio.A competição foi dividida em duas fases. Na primeira fase as oito equipes foram divididas em dois grupos. Na segunda as duas equipes de cada grupo com melhor desempenho se classificaram para uma disputa de semi-final seguida pela disputa de terceiro lugar e pela final.

O evento foi amplamente transmitido por redes de televisão aberta, por emissoras de rádio e pela Internet. Durante o torneio aconteceram 16 partidas. Foram coletados dados durante todos os dias em que ocorreram jogos desse torneio. Entretanto, focamos nossas análises no período entre uma hora antes e uma hora depois da ocorrência de cada jogo. Embora haja coleta de dados fora desse período, o volume de tráfego e o número de clientes é insignificante.

A tabela 1 resume nosso conjunto de dados. Analisamos todas as partidas e verificamos que elas receberam entre aproximadamente 27.000 e 465.000 IPs únicos. A

equipe técnica do Globo.com nos informou que o identificador de sessão foi desabilitado nos servidores nginx devido a problemas de desempenho na maioria dos jogos. Como não é possível identificar sessões de usuários através dos *logs* em todos os jogos (i.e., identificar quais requisições correspondem à exibição de um vídeo para um usuário), consideramos que cada IP seja referente a um cliente. Notamos que esta abordagem pode subestimar a quantidade de usuários assistindo às transmissões em redes com NAT.

A tabela 1 mostra resultados agregados para todo o período monitorado de cada jogo. Além disso, durante a transmissão de um único jogo, observamos até 180.000 espectadores simultâneos. Observamos também mais de 115 Terabytes transferidos durante uma partida e picos de até 350 GB/s. Até onde sabemos, esta quantidade de clientes e volume de tráfego são mais expressivos que os dados analisados anteriormente em outros trabalhos relacionados [Marfia et al. 2007, Jiangchuan Liu and Zhang 2008, Hei et al. 2007, Erman and Ramakrishnan 2013, Shafiq et al. 2013].

Tabela 1. Resumo dos *logs* da Transmissão da Copa das Confederações

Data	Hora	Jogo	IPs Únicos (Milhares)		
			RJ	SP	Total
15/06 - Sáb	16h	Brasil x Japão	93	149	242
16/06 - Dom	16h	México x Itália	38	84	122
16/06 - Dom	19h	Espanha x Uruguai	33	82	115
17/06 - Seg	16h	Taiti x Nigéria	72	114	187
19/06 - Qua	16h	Brasil x México	157	309	465
19/06 - Qua	19h	Itália x Japão	70	112	181
20/06 - Qui	16h	Espanha x Taiti	94	173	267
20/06 - Qui	19h	Nigéria x Uruguai	37	68	105
22/06 - Sáb	16h	Japão x México	10	17	27
22/06 - Sáb	16h	Brasil x Itália	61	110	172
23/06 - Dom	16h	Nigéria x Espanha	33	55	88
23/06 - Dom	16h	Uruguai x Taiti	32	55	87
26/06 - Qua	16h	Brasil x Uruguai	147	293	440
27/06 - Qui	16h	Espanha x Itália	156	230	387
30/06 - Dom	13h	Uruguai x Itália	72	119	190
30/06 - Dom	19h	Brasil x Espanha	126	214	340

3. Caracterização da Carga nos Servidores

Nesta seção, apresentamos os resultados da caracterização da carga de trabalho imposta aos servidores durante as transmissões ao vivo dos jogos da Copa das Confederações. Para analisar a carga nos servidores, caracterizamos a quantidade de conexões atendidas, a taxa de transmissão média e o volume total de dados trafegados pelos servidores.

Analisamos os arquivos de *logs* de todos os jogos. Apenas os quatro primeiros exibiam identificadores de sessão. A figura 4 compara a distribuição acumulada da duração de sessão em um jogo com identificador e a distribuição acumulada do tempo entre chegadas de requisições para listas de reprodução em um jogo sem identificador de sessão. Observamos que ambos os tempos são aproximadamente uniformemente distribuído e menores que 40 segundos em 99% das requisições. Como clientes fazem requisições de listas de reprodução na escala de segundos, esperamos que a carga no servidor e qualidade da mídia recebida pelo cliente não variem significativamente num intervalo de 40 segundos. Além disso, esperamos que cada cliente ativo troque informações com o servidor pelo menos uma vez num intervalo de 40 segundos, e.g., para requisitar a próxima lista de reprodução que contém os próximos segmentos de mídia da transmissão. Usamos essas observações para definir que sessões de usuários terminam após 40 segundos

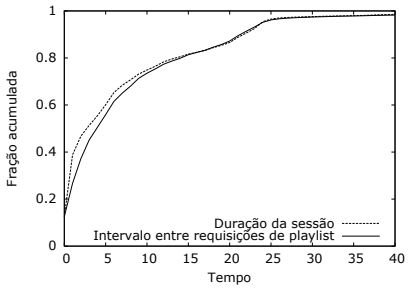


Figura 4. Distribuição do tempo entre requisições de listas de reprodução e duração da sessão

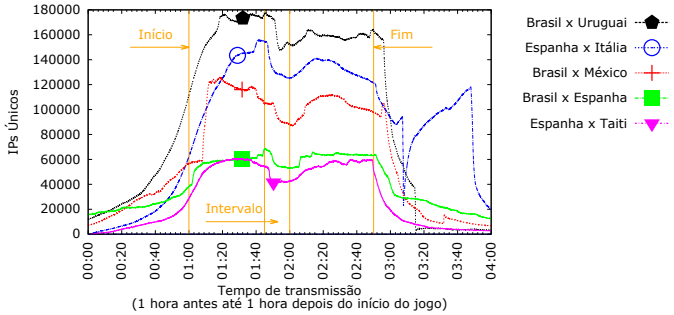


Figura 5. Cinco partidas com maiores pico de acessos simultâneos

sem requisição de listas de reprodução. Por essa razão nós calculamos métricas sobre o estado do sistema num instante de tempo agregando dados em intervalos de 40 segundos (*snapshots*).

Nós categorizamos os jogos de acordo com o pico de carga, calculado como o máximo de sessões de clientes simultâneas (num intervalo de 40 segundos) durante cada jogo. A figura 5 mostra os cinco jogos com os maiores picos de sessões simultâneas. Destacamos na imagem o início, o intervalo e o final das partidas (já considerando períodos de acréscimo típicos e excluindo partidas que tiveram disputa de pênaltis). Dessas cinco partidas, apenas Brasil e Espanha, que fizeram a final do torneio, se enfrentaram em um domingo; todos os outros jogos aconteceram em dias de semana entre 16:00 e 18:00, conforme mostrado na tabela 1.

É possível perceber ainda um maior interesse em jogos do Brasil, o anfitrião e equipe local, e em jogos da Espanha, a então primeira colocada no *ranking* FIFA. Brasil e Espanha tiveram média de 332.000 e 240.000 IPs únicos por partida, respectivamente. Além disso, todos os jogos entre os cinco com maiores números de espectadores envolveram pelo menos uma das duas equipes.

Uma tendência interessante a ser observada é a queda em número de usuários que ocorre entre o primeiro e o segundo tempo. Em todos os cinco jogos a média de espectadores cresce até os 20 primeiros minutos de partida e se mantém relativamente estável até o intervalo do jogo, aos 45 minutos. Da mesma forma, no segundo tempo, a média de usuários cresce nos primeiros 20 minutos e se mantém estável até o final. Por essa razão consideramos que os períodos entre 20 e 45 minutos de cada tempo da partida são os períodos de carga mais estável.

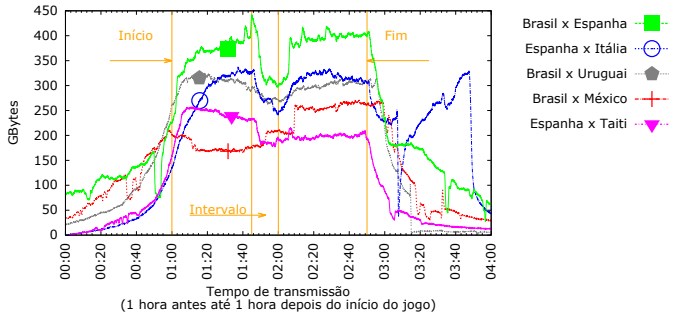


Figura 6. Cinco partidas com maiores picos de volume de transmissão de dados

Nos três jogos de maior carga, há uma queda de, em média, 9% entre o número de clientes conectados no período de pico do primeiro tempo e o número de clientes conectados no pico do segundo tempo. Esse foi o comportamento observado na maioria dos jogos. O jogo entre Brasil e Espanha apresentou um crescimento de aproximadamente 1,8% entre o primeiro e o segundo tempo. Observamos alguns jogos com esse comportamento, e podemos restringir a variação normal no número de clientes do primeiro para o segundo tempo nas partidas de média e alta carga dentro do intervalo de -10% a $+2\%$. Vale a pena destacar ainda que o jogo entre Espanha e Itália foi o único entre os cinco de maior volume no qual aconteceram disputas de pênaltis após a duração regulamentar da partida. Por isso, também foi o único a apresentar uma grande quantidade de acessos na última hora do período analisado.

Analisamos também a carga através do volume de dados trafegados. A figura 6 mostra o volume de tráfego para os cinco jogos com maiores picos de volume de dados transferidos. Observamos que, durante o período de transmissão das partidas mostradas na figura 6, incluindo os intervalos, o tráfego foi bastante intenso. Em particular, a taxa de transferência se aproximou de 450 GB/s e nunca foi menor que 150 GB/s.

Note que a partida com maior volume de tráfego de dados (entre Brasil e Espanha) não foi a mesma partida com maior quantidade de clientes (entre Brasil e Uruguai). Isso aconteceu porque as opções disponíveis de qualidade do streaming de vídeo foram reduzidas. Com vídeos de menor qualidade, o provedor de conteúdo conseguiu atender à demanda não prevista de clientes. A partida entre Brasil e México foi a única que apresentou formato de curva diferente se compararmos as figuras 5 e 6. Apesar do número de clientes no segundo tempo de jogo da partida ter sido menor do que no primeiro, o volume de dados trafegados foi maior.

Estes resultados indicam que a taxa de transmissão varia muito entre jogos e durante um mesmo jogo. Para analisar a qualidade de vídeo e áudio fornecidos pelo Globo.com observamos o tamanho médio das requisições de segmentos de mídia de todas as partidas. Embora a distribuição geral, tenha sido omitida, nós analisamos e mostramos os resultados de alguns jogos específicos a seguir. No geral, observamos que há uma grande variação no tamanho médio de segmentos entre os jogos. Por exemplo, nos dois jogos com maior pico de acessos simultâneos, o tamanho dos segmentos é menor, quando comparado a outros jogos. Consequentemente, a qualidade da transmissão nesses dois eventos foi menor.

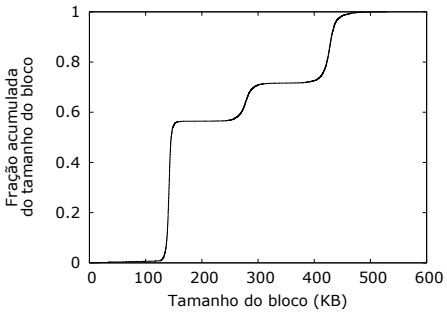


Figura 7. Distribuição do tamanho dos segmentos de mídia do jogo entre Brasil e Uruguai

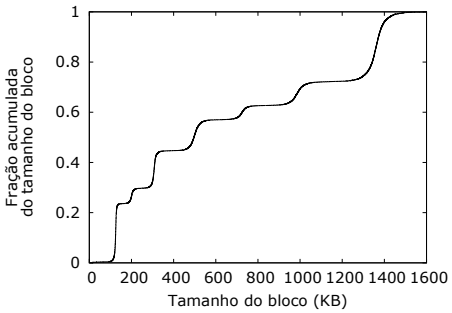


Figura 8. Distribuição do tamanho dos segmentos de mídia do jogo entre Brasil e Espanha

A figura 7 mostra a distribuição do tamanho dos segmentos de mídia transmitidos pelo servidor durante o jogo com maior número de espectadores (entre Brasil e Uruguai). Por essa figura, percebemos que há três platôs bem visíveis, provavelmente correspondendo a três taxas de codificação e níveis de qualidade distintos. Nesse dia, a transmissão com melhor resolução tinha blocos menores que 600 KB. Em contrapartida, no dia do jogo com maior volume de dados trafegados (entre Brasil e Espanha, figura 8) a qualidade do vídeo é maior, com segmentos de até 1600 KB. A figura 8 mostra que, além de maior taxa de codificação, há um número maior de taxas de codificação disponíveis. Devido a variações de desempenho entre diferentes *codecs* de vídeo, é impossível precisar exatamente qual é a qualidade da mídia que está sendo recebida pelos clientes com base somente no tamanho dos segmentos da mídia. Entretanto, como os tempos entre requisições de segmentos sofrem pouca variação (omitido), sabemos que a taxa de transmissão é proporcional ao tamanho dos segmentos.

O tamanho dos segmentos, por sua vez, é função da banda disponível no servidor e da banda disponível na rede do cliente. Esclarecendo, o servidor oferece várias opções de qualidade de mídia e tenta servir sempre a melhor possível. Se não há banda disponível, o servidor deve reduzir a qualidade da mídia e consequentemente o tamanho do bloco de vídeo. Por parte dos clientes, se há disponibilidade de banda na rede, a qualidade de mídia solicitada via DASH é maior, recebendo assim blocos de arquivos maiores.

Além das cinco partidas com maior carga, outra partida que nos chamou a atenção foi a realizada entre Uruguai e Itália, mostrada na figura 9. Essa partida, ao contrário das demais, apresentou crescimento da carga ao longo de toda a duração do jogo. Em uma análise mais profunda, dividimos o tráfego durante a transmissão desta partida recebido pelos dois pontos de distribuição do Globo.com. Nosso intuito foi de verificar se houve alguma anomalia em algum ponto de distribuição que pudesse impactar na análise como um todo. No entanto, o comportamento foi o mesmo nos dois pontos de distribuição de mídia ao vivo. Ressaltamos que essa partida foi a única realizada em um domingo às 13 horas. Portanto, conjecturamos que seu comportamento anormal se deve a essa diferença no horário ou dia de transmissão. Infelizmente, devido à curta duração e poucas partidas realizadas durante o campeonato, não há parâmetros de comparação que nos permitam confirmar essa especulação. Notamos que no jogo entre Uruguai e Itália houve disputa de pênaltis, por isso a carga no servidor continua aumentando após o término do jogo.

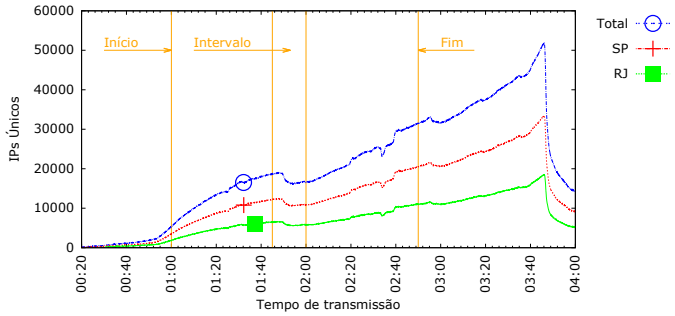


Figura 9. Quantidade de usuários total e por localização do servidor da partida entre Uruguai e Itália

Ainda na figura 9 mostramos que o número de clientes conectados aos servidores localizados no Rio de Janeiro é significativamente menor do que nos servidores localizados em São Paulo. Essa diferença aconteceu em todas as partidas, variando entre 50% e 60% de carga extra em São Paulo. É importante observar essa discrepância, uma vez que no planejamento atual do sistema, informado pelo Globo.com, a quantidade de banda disponível nos pontos de distribuição é praticamente a mesma.

4. Caracterização dos Clientes

Nesta seção caracterizamos a distribuição dos clientes entre os principais provedores de acesso à Internet no Brasil, correlacionando isto com a qualidade da mídia recebida, região geográfica e indicadores sócio-econômicos. Procuramos analisar o comportamento da rede e a distribuição dos usuários por provedores de acesso à Internet durante a transmissão das partidas. Para isso observamos os principais sistemas autônomos (ASes) correspondentes aos endereços IP dos espectadores. Para mapear endereços IP em sistemas autônomos usamos a base GeoIP da Maxmind [2].

As figuras 10 e 11 mostram a quantidade de clientes nos três provedores com maior número de clientes assistindo a partida entre Brasil e Uruguai. Dividimos as figuras entre clientes conectados aos servidores do Rio de Janeiro e São Paulo, respectivamente. É possível observar que no Rio de Janeiro há um amplo domínio da Telemar, com cerca de 45% dos clientes. Em São Paulo a divisão é mais próxima, mas a Global Village Telecom (GVT) lidera com cerca de 27% dos clientes. Analisando os dados como um todo verificamos que, em São Paulo, as três empresas responsáveis pelos sistemas autônomos mostrados atendem a cerca de 70% de todo o tráfego da região. No Rio de Janeiro, esse percentual é ainda maior, chegando a 80%. Essa concentração de acesso reflete uma grande dependência de poucos provedores de acesso. Caso haja falha em algum dos seis ISPs citados neste trabalho, a provedora de conteúdo perderia uma quantidade relevante de seus clientes. Em um cenário ainda pior, falhas em mais de um desses provedores poderiam levar à indisponibilidade regional generalizada.

Com intuito de analisar a qualidade da mídia recebida pelos clientes dos principais provedores, analisamos a partida entre Brasil e Espanha, que foi a que apresentou maior volume de dados trafegados e uma variação maior de taxas de codificação da mídia (figura 8). Os resultados dessa análise podem ser vistos na figura 12. É possível perceber

[2]http://www.maxmind.com/

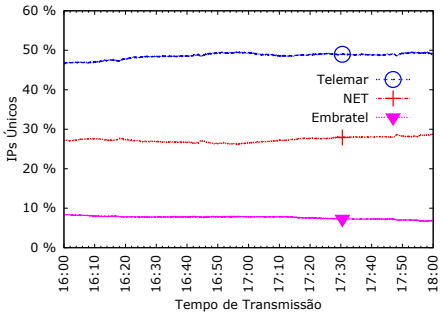


Figura 10. Três ASs com maior número de usuários nos servidores do Rio de Janeiro

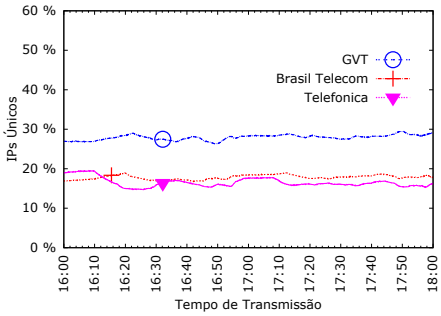


Figura 11. Três ASs com maior número de usuários nos servidores de São Paulo

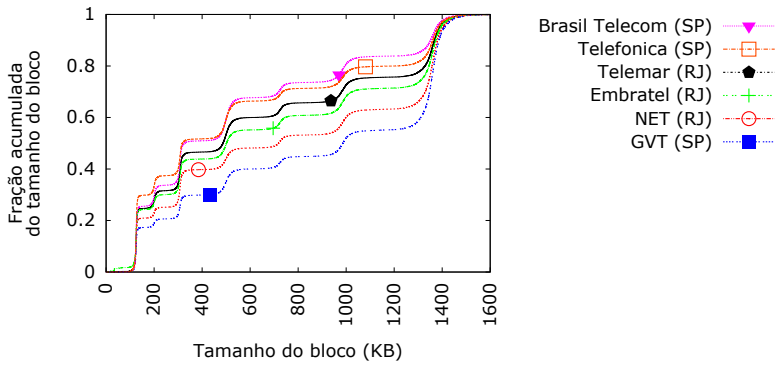


Figura 12. Distribuição do tamanho dos segmentos de mídia por provedores no jogo Brasil e Espanha

que GVT, NET e Embratel, em geral, recebem blocos maiores de segmentos de mídia se comparados a Telemar, Brasil Telecom e Telefônica. Isso parece indicar que os clientes dos primeiros provedores recebem um conteúdo de melhor qualidade do que os clientes dos outros provedores. Note que no Rio de Janeiro, embora a Telemar agrupe a maior quantidade de clientes, a qualidade recebida é inferior à dos clientes atendidos por Embratel e NET. Em São Paulo estão os provedores com maior e menor média de tamanho do bloco, respectivamente GVT, com média de aproximadamente 875 KB e Brasil Telecom, com média de aproximadamente 560 KB. Apesar de não termos informações para tirar conclusões sobre estas observações, elas motivam estudos sobre diversidade de rotas e provisionamento de capacidade nas redes dos provedores de acesso brasileiros.

4.1. Caracterização Regional

Encerramos nossa caracterização apresentando um mapa regional da qualidade da mídia distribuída para diferentes regiões do Brasil. Usamos as bases de dados GeoIP e GeoIP-City da Maxmind para identificar em quais regiões estavam localizados os IPs dos clientes do Globo.com. Também utilizamos dados de população e índice de desenvolvimento humano (IDH) disponibilizados pelo IBGE [3].

A figura 13 mostra a mediana e os quartis do tamanho dos blocos de mídia recebidos por clientes agrupados em cada estado do Brasil. Ordenamos os estados no eixo x

[3] http://www.ibge.gov.br/estadosat/

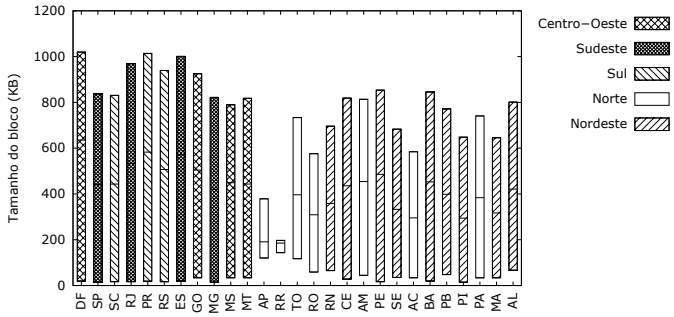


Figura 13. Tamanho do bloco de mídia por estado ordenado por IDH.

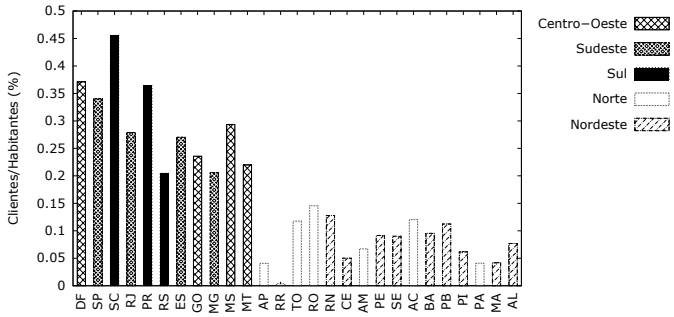


Figura 14. Relação Espectadores/habitantes por estado ordenado por idh

pelo IDH em ordem decrescente. Observamos que à exceção do estado de Roraima, do qual a amostra de IPs foi muito pequena, todas as regiões apresentaram grande variação, mas nós observamos uma tendência geral de regiões com maior IDH terem maiores tamanhos de blocos. Isso é um indicador de que o acesso à Internet em banda larga é de melhor qualidade nessas regiões. A região Nordeste apresenta distribuição heterogênea, tendo estados com variados perfis de qualidade. Por fim, a região Norte é a que possui indicadores de qualidade de acesso em banda larga mais baixos.

Usamos o número de espectadores para tentar estimar a penetração da banda larga no Brasil. A figura 14 mostra uma relação percentual entre espectadores e habitantes de cada estado. Novamente ordenamos os estados no eixo x pelo IDH em ordem decrescente. Verificamos maior relação espectadores/habitantes nos estados das regiões Centro-Oeste, Sul e Sudeste bem como forte relação com o IDH. Essa relação também é observada na pesquisa nacional por amostra de domicílios (PNAD), realizada pelo IBGE em 2011[4] e pode indicar maior penetração de acesso à banda larga nessas regiões.

Aprofundamos a caracterização regional focando em cidades de três estados da região sudeste que apresentaram médias de tráfegos distintas entre suas capitais e algumas cidades pólo. A tabela 2 mostra os resultados dessa análise. Em geral, as cidades com mais habitantes receberam blocos maiores do que as cidades com menos. Observamos, nos estados do Rio de Janeiro e São Paulo, uma grande concentração de clientes nas capitais. Acreditamos que isso se deve à extensão das regiões metropolitanas dessas cidades. O estado de Minas Gerais apresentou distribuição geográfica dos clientes menos discrepante, com menor concentração na capital. Embora logicamente as cidades com

[4]http://www.ibge.gov.br/home/estatistica/populacao/acessoainternet2011/default.shtm

Tabela 2. Análise do tráfego por cidade de MG, RJ e SP

Estado	Cidade	População	IPs únicos	IPs/População(%)	Tamanho médio do bloco(KB)
MG	Belo Horizonte	2.375.151	14.012	0,58	620
	Uberlândia	654.681	3.990	0,61	482
	Juiz de Fora	550.710	2.272	0,41	522
	Montes Claros	390.212	575	0,15	348
RJ	Rio de Janeiro	6.320.446	29.049	0,46	663
	Petrópolis	298.017	403	0,14	500
	Volta Redonda	262.259	214	0,08	545
	Macaé	229.624	497	0,22	476
SP	São Paulo	11.895.893	54.256	0,46	585
	Campinas	1.154.617	6.586	0,57	598
	Ribeirão Preto	658.059	3.048	0,46	524
	Sorocaba	637.187	2.116	0,33	613

maiores populações sejam aquelas com mais clientes, não encontramos proporção clara entre o número de habitantes e a quantidade de espectadores.

5. Trabalhos Relacionados

Com o crescimento da Internet, também observamos um crescimento em sua complexidade. Eventos antes bem compreendidos podem não corresponder à realidade atual, face a escala que estes podem tomar. Por exemplo, há menos de 10 anos, grandes eventos de mídia ao vivo alcançavam cerca de 3,5 milhões de requisições de usuários em um período de quatro semanas [Veloso et al. 2002]. Hoje em dia, uma única transmissão, como a final de um campeonato de futebol, ou o discurso de um presidente, pode atrair a atenção de dezenas de milhões de usuários, em menos de uma hora[5].

Um dos primeiros trabalhos a caracterizar transmissões de vídeo ao vivo na Internet foca no comportamento dos usuários que assistiam as transmissões [Veloso et al. 2002]. Especificamente, os autores analisaram a transmissão de um canal ao vivo na Internet e modelaram características dos seus usuários, como por exemplo, o tempo que eles permaneciam assistindo a transmissão ao vivo. Diferentemente daquele trabalho, em nosso trabalho estamos interessados em características do tráfego e seu impacto.

Grande parte das soluções para transmissão ao vivo na Internet utilizam distribuição em redes com arquitetura descentralizada (i.e., redes Par-a-Par, ou P2P) para melhorar sua escalabilidade. Por consequência, há uma série de trabalhos que avaliam tais sistemas e seus usuários [Hei et al. 2007, Jiangchuan Liu and Zhang 2008, Marfia et al. 2007, Chen et al. 2013]. Nessa linha, os autores focam no comportamento dos usuários e no desempenho da rede. Alguns desses trabalhos mostram que existem eventos de grande interesse que foram transmitidos em arquitetura P2P. Em alguns casos, os sistemas atenderam quase 200 mil usuários simultâneos [Hei et al. 2007]. O sistema que avaliamos não utiliza distribuição P2P, o que aumenta os problemas de escalabilidade e motiva estudos de caracterização de carga para o adequado provisionamento de capacidade.

Mais similar ao nosso trabalho, [Erman and Ramakrishnan 2013] fazem uma caracterização da transmissão ao vivo do Superbowl, o maior evento esportivo dos Estados Unidos. Porém, ao contrário do nosso trabalho, eles avaliam apenas a parcela do tráfego que passa por uma rede de telefonia celular.

Assim, apesar de existir uma série de trabalhos que estudam, caracterizam e mo-

[5]http://gigaom.com/2009/02/07/cnn-inauguration-p2p-stream-a-success-despite-backlash/

delam alguns aspectos de transmissão de vídeo ao vivo na Internet, poucos focam no impacto gerado por um grande evento. Mais ainda, a maioria dos trabalhos é voltada para a análise do comportamento dos usuários. Neste trabalho nós focamos na carga imposta aos servidores e na qualidade de serviço aos clientes. Outros trabalhos enfocam a caracterização e modelagem do comportamento de usuários em sistemas de transmissão sob demanda de vídeo pré-armazenado. Por exemplo, alguns estudos abordaram propriedades do comportamento interativo destes usuários [Costa et al. 2004] e [Rocha et al. 2005]. Os autores avaliam os perfis de uso, com foco na interatividade do sistema por parte dos usuários, caracterizando por exemplo, o tempo de permanência e o padrão de execução das mídias. Transmissões ao vivo, foco do trabalho atual, têm características inerentemente diferentes de transmissões sob demanda. Logo, consideramos os esforços nesta direção complementares ao nosso.

6. Conclusões e trabalhos futuros

Transmissão de vídeo ao vivo na Internet é cada vez mais comum. Grandes eventos são difundidos ao vivo e, potencialmente, atraem um grande número de espectadores. Esse novo horizonte, com transmissões que, de repente, atraem um número de clientes acima do esperado impõe desafios para provisionamento de capacidade de provedores de serviço na Internet. Neste trabalho apresentamos uma caracterização da carga de trabalho de servidores do Globo.com durante transmissões ao vivo dos jogos da Copa das Confederações 2013. Nós avaliamos a arquitetura de serviços da Globo.com que, em alguns jogos, suporta carga muito elevada. Por exemplo, as transmissões avaliadas atendem até 180 mil sessões simultâneas e disseminam até 115 TB de dados em um único jogo com taxas de transmissão instantâneas que chegam a 350 GB/s.

Além disso, o número de espectadores, e consequentemente a carga imposta aos servidores, varia significativamente (1600%) entre os jogos com maior e menor carga. Isso ocorre, principalmente, em função da popularidade dos times e do horário da transmissão. Essa variação de carga pode levar a problemas de provisionamento. Por exemplo, mostramos que houve uma limitação na qualidade da mídia para adequar o consumo de banda e o número inesperado de clientes aos recursos existentes.

Caracterizamos também a qualidade da mídia recebida pelos usuários em função de seus provedores de acesso à Internet. Observamos que, grandes provedores de Internet banda larga no Brasil direcionam grande parte de seus clientes para o mesmo ponto de distribuição do Globo.com. Espectadores assinantes do maior provedor de acesso à Internet avaliado foram os que receberam mídia na menor qualidade. Também avaliamos a qualidade da mídia recebida pelos usuários em função de sua localização geográfica e IDH do estado. Nossos resultados mostram que a qualidade da mídia e a fração da população assistindo ao jogo pela Internet são correlacionadas com o IDH.

Em suma, nossos resultados confirmam os desafios para o provisionamento de capacidade em servidores de vídeo durante grandes eventos. Em trabalhos futuros pretendemos estudar técnicas para prever a carga durante grandes eventos. Também pretendemos avaliar o impacto da transmissão de grandes eventos ao vivo na infra-estrutura de rede, correlacionando a carga nos servidores com medições de rota obtidas com *traceroute*. Por último, estamos em contato com o Globo.com para recuperar e avaliar a carga da Copa do Mundo de Futebol 2014, também realizada no Brasil.

7. Agradecimentos

Os autores agradecem o apoio do CNPq, CAPES e da FAPEMIG e também a ajuda dos times de tecnologia da globo.com.

Referências

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`.

Chen, Y., Zhang, B., Liu, Y., and Zhu, W. (2013). Measurement and modeling of video watching time in a large-scale internet video-on-demand system. *Multimedia, IEEE Transactions*, 15(8):2087–2098.

Costa, C. P., Cunha, I. S., Borges, A., Ramos, C. V., Rocha, M. M., Almeida, J. M., and Ribeiro-Neto, B. (2004). Analyzing client interactivity in streaming media. In *Proceedings of the 13th international conference on World Wide Web*, pages 534–543. ACM.

Erman, J. and Ramakrishnan, K. (2013). Understanding the super-sized traffic of the super bowl. *IMC '13 Proceedings of the 2013 conference on Internet measurement conference*, pages 353–360.

Hei, X., Liang, C., Liang, J., Liu, Y., and Ross, K. W. (2007). A measurement study of a large-scale p2p iptv system. *IEEE Transactions Multimedia*, pages 1672–1687.

Jiangchuan Liu, Sanjay G. Rao, B. L. and Zhang, H. (2008). Opportunities and challenges of peer-to-peer internet video broadcast. *Proceedings of the IEEE*, pages 11–24.

Katabi, D. and Wroclawski, J. (2000). A Framework for Scalable Global IP-anycast. In *Proc. ACM SIGCOMM*.

Mahanti, A. (2014). The evolving streaming media landscape. *Internet Computing, IEEE*, 18(1):4–6.

Marfia, G., Sentivelli, A., Tewari, S., Gerla, M., and Kleinrock, L. (2007). Will IPTV ride the peer-to-peer stream? In *Communications Magazine, Special Issue on Peer-to-Peer Streaming*. IEEE.

Rocha, M., Maia, M., Cunha, Í., Almeida, J., and Campos, S. (2005). Scalable media streaming to interactive users. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 966–975. ACM.

Shafiq, M. Z., Ji, L., Liu, A. X., Pang, J., Venkataraman, S., and Wang, J. (2013). A first look at cellular network performance during crowded events. *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 17–28.

Stockhammer, T. (2011). Dynamic adaptive streaming over http – standards and design principles. *MMSys '11 Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144.

Veloso, E., Almeida, V., Meira, W., Bestavros, A., and Jin, S. (2002). A hierarchical characterization of a live streaming media workload. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 117–130. ACM.

Uma Avaliação do Ataque de Negação de Serviço em Conluio Consumidor-Produtor em Redes Orientadas a Conteúdo

André Luiz Nasserala Pires[1,2], **Igor Monteiro Moraes**[1]

[1]Laboratório MídiaCom, PGC-TCC
Instituto de Computação - Universidade Federal Fluminense
Niterói, Rio de Janeiro, Brasil

[2]Centro de Ciências Exatas e Tecnológicas, CCET
Universidade Federal do Acre
Rio Branco, Acre, Brasil

{anasserala,igor}@ic.uff.br

Abstract. *This paper evaluates a denial-of-service attack in information-centric networks based on the Content Centric Networking (CCN) architecture. This attack aims at increasing the content retrieval time. In this attack, both malicious consumers and producers collude, by generating, publishing and changing content popularity. Malicious contents are stored by intermediate nodes and occupy the cache space that should be occupied by legitimate content. Thus, the probability of a legitimate consumer retrieves content directly from the original producer increases as well as the content retrieval time. We evaluate the impact of the attack by varying the number of consumers and producers in collusion, the interest packets rate and the way malicious contents are requested. Results show if 20% of consumers are malicious and send 500 interests/s each, the content retrieval time experienced by legitimate users increases by 20 times, which shows the effectiveness of the attack.*

Resumo. *Este artigo avalia um ataque de negação de serviço em redes orientadas a conteúdo baseadas na arquitetura Content Centric Networking (CCN), cujo objetivo é aumentar o tempo de recuperação de conteúdos. Nesse ataque, consumidores e produtores maliciosos agem em conluio, gerando, disponibilizando e manipulando a popularidade de conteúdos. Esses conteúdos maliciosos são armazenados pelos nós intermediários e ocupam o espaço em cache que deveria ser ocupado por conteúdos legítimos. Assim, aumenta-se a probabilidade de um consumidor legítimo recuperar o conteúdo diretamente do produtor original, aumentando o seu tempo de recuperação. Avalia-se o impacto do ataque em função do número de consumidores e produtores em conluio, da taxa de pacotes de interesse e do padrão de solicitações de conteúdos maliciosos. Resultados mostram que se 20% dos consumidores são maliciosos e enviam 500 interesses/s cada um, o tempo de recuperação de conteúdos por consumidores legítimos aumenta em 20 vezes, o que evidencia a efetividade do ataque.*

1. Introdução

As Redes Orientadas a Conteúdo são um novo paradigma de comunicação para Internet [Brito et al. 2012]. O principal objetivo dessas redes é a entrega de conteúdo

para os usuários independentemente da localização desse conteúdo, ao contrário da arquitetura TCP/IP, cujo objetivo é a comunicação entre sistemas finais. Diversas arquiteturas foram propostas para esse novo paradigma de comunicação e uma das arquiteturas com maior destaque na literatura é a *Content Centric Networking* (CCN) [Jacobson et al. 2009, Smetters e Jacobson 2009]. Entre as principais características da CCN estão o roteamento através de nomes de conteúdo, o armazenamento de conteúdo em nós intermediários da rede e a capacidade de auto-certificar o conteúdo, aplicando a segurança diretamente aos pacotes de dados [Jacobson et al. 2009].

Uma das mais vantajosas características da CCN é o consumo indireto de conteúdo, ou seja, qualquer nó da rede que ao receber uma solicitação de conteúdo e possua esse conteúdo em *cache* pode enviar tal conteúdo para o nó solicitante. Na CCN, o nó que solicita o conteúdo é chamado de consumidor e qualquer nó que disponibilize o conteúdo é chamado de produtor. O produtor original é a fonte de um conteúdo. Portanto, na CCN, é possível que um determinado nó, que esteja mais próximo do consumidor, consiga responder à solicitação de um conteúdo sem que o consumidor seja obrigado a recuperar esse conteúdo diretamente do produtor original, que pode estar mais distante. Assim, o tempo de recuperação de conteúdos pode ser reduzido. Além disso, o armazenamento de conteúdo em *cache* aumenta a disponibilidade de conteúdos e pode reduzir o consumo de banda, uma vez que o conteúdo é encaminhado por menos saltos.

Outra característica da CCN é que a segurança é aplicada diretamente aos conteúdos, diferentemente da arquitetura TCP/IP, na qual a segurança é aplicada ao canal de comunicação entre os sistemas finais [Smetters e Jacobson 2009]. Um pacote CCN é auto-certificado, isto é, ele contém a assinatura digital do pacote e a chave pública do produtor [Jacobson et al. 2009]. Portando, é possível verificar a integridade do pacote e se ele foi gerado pelo produtor que possui tal chave pública. As assinaturas geram um *overhead* tanto para o produtor assinar o conteúdo quanto para os consumidores verificarem. Além disso, a CCN é mais robusta a ataques de negação de serviço (*Denial of Service* - DoS) comuns na Internet atual, como o de esgotamento de banda e o de reflexão, em virtude do uso de *cache* pelos nós intermediários e da agregação de solicitações de conteúdo [Gasti et al. 2013], como será discutido na Seção 2.

Um ataque de negação de serviço particular, chamado de conluio produtor-consumidor, entretanto, pode ser efetivo porque os mecanismos nativos empregados pela CCN não são suficientes para inibi-lo. Não foi encontrado na literatura um tipo de ataque onde produtores e consumidores agem em conluio no âmbito de redes CCN. Este trabalho avalia o impacto desse ataque em redes CCN, cujo objetivo é aumentar o tempo de recuperação de conteúdos. Nesse ataque, consumidores maliciosos solicitam conteúdos que são disponibilizados apenas por produtores maliciosos a uma alta taxa. Isso aumenta o tempo de recuperação de conteúdos legítimos, em virtude do aumento da taxa de erro do *cache* (*cache miss*) para esses conteúdos e, consequentemente, da necessidade de nós legítimos terem que recuperar o conteúdo diretamente do seu produtor original. Os mecanismos de segurança da CCN padrão são ineficazes na detecção do ataque em conluio, pois, do ponto de vista da rede, as solicitações e os conteúdos são legítimos. São enviados pacotes de interesse para conteúdos que existem e que são disponibilizados por produtores. O conteúdo é malicioso porque torna popular um conteúdo que não é de interesse de usuários legítimos. Como o produtor malicioso assina os conteúdo de acordo

com a política definida pela CCN, os consumidores maliciosos podem solicitá-los sem risco de que esses conteúdos sejam descartados por mecanismos de verificação de assinaturas e chaves. Esse ataque é possível, porque a CCN emprega políticas de substituição do *cache* baseadas, em sua maioria, na popularidade dos conteúdos. Assim, se um determinado conteúdo não é solicitado com frequência ou não foi solicitado recentemente pelos consumidores, ele é considerado menos popular. Dessa forma, esse conteúdo terá prioridade de descarte quando houver necessidade de armazenar novos conteúdos. Vários nós maliciosos podem, então, solicitar um conjunto específico de conteúdos produzidos maliciosamente e em taxas altas de envio de interesse para manipular a política de *cache*. Assim, dependendo da forma como os conteúdos maliciosos são solicitados, é possível até remover conteúdos legítimos do *cache*.

A avaliação do ataque de conluio consumidor-produtor é feita através de simulações para diferentes configurações, nas quais se variam o número de consumidores e produtores em conluio, a taxa de pacotes de interesse e o padrão de solicitações de conteúdos maliciosos. As métricas empregadas são o tempo de recuperação de conteúdos legítimos, o percentual de ocupação maliciosa do *cache*, o percentual da taxa de erros de *cache* de conteúdos legítimos e o percentual de conteúdos legítimos recuperados do produtor original. Os resultados mostram que o ataque compromete uma das maiores vantagens das CCN que é a redução do tempo de recuperação de conteúdos pelo uso do *cache* nos nós intermediários. Conclui-se que se 20% dos nós consumidores são maliciosos e enviam 500 solicitações de interesses por segundo cada um, o tempo de recuperação de conteúdos por usuários legítimos aumenta em cerca de 20 vezes na topologia avaliada. Além disso, observa-se que até 67% dos conteúdos legítimos são recuperados diretamente do produtor original nas configurações analisadas. .

O restante do artigo está organizado da seguinte forma. Na Seção 2 uma revisão sobre o funcionamento e aspectos de segurança da CCN é apresentada. Na Seção 3 os trabalhos relacionados são discutidos. Na Seção 4, o ataque de negação de serviço em conluio consumidor-produtor é descrito. Na Seção 5, é definido o cenário de avaliação usado nas simulações. Na Seção 6, os resultados dos experimentos são analisados e discutidos. Por fim, na Seção 7, as conclusões são apresentadas.

2. A Arquitetura CCN: Funcionamento e Aspectos de Segurança

A arquitetura CCN tem como objetivos aumentar a disponibilidade e reduzir o tempo de recuperação de conteúdos. Na CCN, os nós da rede possuem um *cache* para armazenar conteúdos recebidos previamente. Consequentemente, qualquer nó pode responder a um pedido, se o conteúdo solicitado está disponível em seu *cache*, conhecido como *Content Store* (CS). Esses nós são chamados de produtores e os consumidores são os nós que solicitam um conteúdo. Quanto mais produtores de um conteúdo na rede, maior a disponibilidade desse conteúdo e maior a probabilidade de consumidores recuperarem um conteúdo de um produtor mais próximo. Essa é uma das vantagens da CCN em comparação com a arquitetura atual da Internet.

A CCN emprega dois tipos de pacotes: interesse e dados. Consumidores enviam pacotes de interesse para solicitar um conteúdo. Os produtores respondem aos interesses com pacotes de dados, que carregam o conteúdo em si ou pedaços de conteúdo [Brito et al. 2013]. Os nós encaminham tanto pacotes de interesse quanto pa-

cotes de dados com base no próprio nome do conteúdo, ao invés do endereço de destino do nó que possui o conteúdo. Para realizar o encaminhamento de pacotes, cada nó CCN tem duas estruturas de dados: a *Pending Interest Table* (PIT) e a *Forwarding Information Base* (FIB). A PIT guarda o estado de cada pacote de interesse encaminhado por um nó que ainda não recebeu uma resposta, ou seja, os interesses que esperam por um pacote de dados. Cada entrada da PIT também armazena a interface de recepção de um pacote de interesse. É importante ressaltar que o tamanho da PIT é limitado e, dessa forma, novos interesses que chegam enquanto a tabela está cheia não são encaminhados. Esse fato é explorado por usuários maliciosos, como detalhado nos próximos parágrafos. A FIB atua como uma tabela de encaminhamento para pacotes de interesse. Essa tabela contém uma lista de entradas, cada uma contendo o prefixo de um nome e uma lista de interfaces de saída para as quais os pacotes de interesse com nomes de mesmo prefixo devem ser encaminhados.

Quando um nó CCN recebe um pacote de interesse, ele verifica seu CS para encontrar uma cópia do conteúdo solicitado, cujo nome está no cabeçalho do pacote de interesse. Se o conteúdo está armazenado em *cache*, o nó envia um pacote de dados para o consumidor. Caso contrário, o nó verifica a sua PIT. Se houver uma entrada na PIT para o mesmo conteúdo, o nó atualiza a lista de interfaces de entrada e descarta o pacote de interesse. Esse procedimento é chamado de agregação de pacotes de interesse e torna a CCN mais robusta contra ataques de DoS, como discutido a seguir. Caso contrário, o nó cria uma nova entrada na PIT e, então, consulta a FIB para determinar a interface de saída para encaminhar o pacote de interesse. Se não houver nenhuma entrada na FIB relacionada com o nome do conteúdo, o pacote de interesse é enviado em todas as interfaces de saída. Os nós repetem este processo de encaminhamento para cada pacote de interesse recebido. Os pacotes de dados seguem o caminho reverso percorrido pelos pacotes de interesse porque a PIT armazena a lista de interfaces com interesses pendentes a serem atendidos [Zhang et al. 2010].

Ataques de negação de serviço (DoS) são uma ameaça na Internet atual. A arquitetura CCN, entretanto, é mais robusta a esse tipo de ataque do que a pilha TCP/IP em virtude de duas características: o armazenamento de conteúdo pelos nós intermediários e a agregação de pacotes de interesse [Gasti et al. 2013]. Ataques de esgotamento de banda e de reflexão, por exemplo, são pouco eficientes na CCN.

Ataques de esgotamento de banda inundam a vítima com requisições de serviço para esgotar seus recursos. Neste ataque, os pacotes devem chegar à vítima para que o ataque seja efetivo. Na CCN, no entanto, os pacotes não possuem o endereço de destino e os consumidores não podem garantir que os pacotes de interesse alcancem a origem do conteúdo, ou seja, o produtor original e nesse a caso a vítima, porque qualquer nó pode responder ao interesse. Portanto, os consumidores maliciosos podem gerar uma quantidade enorme de pacotes de interesse para um dado conjunto de conteúdos, mas nenhum ou poucos desses pacotes alcançarão o produtor original.

É importante ressaltar também que nós no caminho reverso entre o consumidor e o produtor armazenam conteúdos em *cache*. Assim, nós intermediários entre consumidor e produtor provavelmente irão satisfazer novos pacotes de interesse, que dificilmente alcançarão o produtor, dependendo da popularidade destes conteúdos. Além disso, a CCN reduz o número de pacotes de interesse transmitidos. Um nó só envia um pacote de in-

teresse que não corresponde a uma entrada PIT. Caso contrário, o nó atualiza a lista de interfaces e descarta o pacote, como descrito nos parágrafos anteriores.

Ataques de reflexão são baseados na técnica de falsificação de endereços IP (*IP spoofing*) e visam atacar vítimas diferentes simultaneamente. Na CCN, esses ataques são menos eficazes porque os pacotes de dados são sempre encaminhados para o consumidor através do caminho reverso percorrido pelo pacote de interesse. Consumidores também não podem garantir que os pacotes de interesse cheguem às vítimas intermediárias ou finais devido ao *cache* nos nós intermediários. Nós CCN, porém, enviam pacotes de interesse em todas as suas interfaces, se não houver nenhuma entrada na FIB para um prefixo de nome solicitado. Portanto, se o atacante e a vítima estão na mesma sub-rede do ataque, a reflexão pode ser eficaz [Gasti et al. 2013]. Neste cenário, o atacante pode enviar pacotes de interesse através de todas as suas interfaces com os endereços da camada MAC falsificados. Assim, múltiplas cópias do conteúdo são enviadas para a vítima. Para evitar isso, nós CCN não transmitem o mesmo conteúdo mais de uma vez no mesmo domínio de difusão (*broadcast*) [Gasti et al. 2013].

Apesar de ser mais robusta do que a arquitetura TCP/IP aos ataques de DoS atuais, a arquitetura CCN possui ataques e vulnerabilidades identificados em trabalhos recentes [Gasti et al. 2013, Ribeiro et al. 2012], que são discutidos na Seção 3.

3. Trabalhos Relacionados

Os ataques de negação de serviço em CCN são classificados em dois tipos: ataques por inundação de interesses ou envenenamento de *cache* [Ribeiro et al. 2012].

O objetivo dos ataques de inundação de interesses é sobrecarregar a PIT com solicitações de conteúdo enviadas por um nó malicioso a uma alta taxa [Guimarães et al. 2013]. Os pacotes de interesse maliciosos, em geral, solicitam conteúdos inexistentes, o que mantém por mais tempo a informação sobre esses interesses na PIT de um nó. A informação sobre um interesse pendente só é removida após o estouro de um temporizador. Enquanto aguarda pelo pacote de dados, o nó receberá novos interesses para outros conteúdos inexistentes. No pior caso, com a PIT cheia, um nó afetado não atenderá interesses legítimos, o que leva à queda de desempenho da rede.

Gasti *et al.* definem o ataque de inundação de interesses e propõem um mecanismo de *push-back* como contramedida [Gasti et al. 2013]. Esse mecanismo monitora a ocupação da PIT e identifica quando uma determinada interface está próxima de atingir seu número máximo de entradas na PIT. Assim, o mecanismo controla o fluxo de pacotes de interesse que contém os mesmos prefixos de nome. Além disso, a contramedida envia uma notificação na interface supostamente atacada que será recebida por um nó vizinho. Esse nó, por sua vez, deve propagar tal informação no sentido das interfaces atacadas e, ao mesmo tempo, limitar a taxa de interesses encaminhados que contenham o prefixo sob ataque. Portanto, o objetivo da contramedida é empurrar o ataque para o caminho de volta até o atacante, ou pelo menos para um nó no qual seja detectado [Gasti et al. 2013]. A principal característica dessa contramedida é não modificar a arquitetura padrão proposta para a CCN. O ponto fraco do trabalho de Gasti *et al.* é que nem o impacto do ataque e nem a contramedida proposta são avaliados por simulação ou experimentos práticos.

Choi *et al.*, por outro lado, avaliam através de simulações a efetividade do ataque de inundação de interesses [Choi et al. 2013]. Os autores mostram que em uma rede com

poucos nós, o desempenho é comprometido. Conclui-se que a vazão de dados total de consumidores legítimos diminuiu cerca de 65%. Da mesma forma, observa-se que o tempo médio de recuperação de conteúdos aumenta rapidamente, logo após o início do ataque.

Afanasyev *et al.* também avaliam o ataque de inundação de interesse através de simulações, porém consideram diferentes cenários e uma rede de maior escala do que a usada no trabalho anterior [Afanasyev et al. 2013]. Os autores também avaliam a contramedida baseada em um mecanismo de *push-back* proposta por Gasti *et al.*. Os resultados mostram que essa contramedida é eficiente, pois isola por completo os atacantes de modo que eles causem pouco ou nenhum impacto no desempenho percebido por usuários legítimos.

Diferentemente dos ataques de inundação de interesses, o objetivo do ataque de envenenamento de *cache* é ocupar o *cache* dos nós com conteúdo poluído. Esse conteúdo é enviado por consumidores maliciosos para fazer com que nós armazenem um conteúdo que possua uma assinatura válida, porém corrompido ou aumentem a popularidade de conteúdos menos populares. No primeiro caso, o objetivo é reduzir o espaço disponível em *cache* para armazenar conteúdos legítimos e fazer com que consumidores recebam conteúdos corrompidos. No segundo, o objetivo é remover do *cache* conteúdos legítimos assumindo que uma política de substituição baseada na popularidade dos conteúdos é usada. Uma contramedida ao ataque de envenenamento de *cache* é a verificação da assinatura contida nos pacotes de dados [Zhang et al. 2010]. Por padrão, a assinatura dos conteúdos é verificada apenas pelos nós de borda, ou seja, os consumidores, e não pelos nós intermediários da rede. Essa característica garante que os consumidores não recebam pacotes de dados contendo conteúdo malicioso. Nesse caso, o serviço da CCN é negado se os consumidores sempre receberem conteúdos inválidos. A solução de obrigar a verificação da assinatura de todos os conteúdos em todos os nós implica sobrecarga de processamento e, por isso, é de difícil adoção prática [Gasti et al. 2013].

Ribeiro *et al.* propõem um mecanismo de verificação probabilística de assinaturas [Ribeiro et al. 2013, Ribeiro et al. 2014]. O mecanismo proposto é eficiente, porém se mostrou dependente da topologia de rede utilizada. Quanto maior o número de saltos, maior a probabilidade do conteúdo poluído ser descartado ao longo do caminho. Outra proposta similar, chamada CacheShield, também usa dados estatísticos para verificar se um conteúdo é poluído ou não [Xie et al. 2012], porém tem as mesmas limitações do trabalho de Ribeiro *et al.*.

Kim *et al.* investigam o impacto de fluxos de conteúdo de longa duração na CCN [Kim et al. 2013]. A presença de fluxos de longa duração pode ter efeito similar ao ataque de envenenamento de *cache*. Se fluxos de longa duração ocuparem temporariamente um *cache* de um nó por determinado conteúdo, eles podem expulsar pedaços de conteúdos populares do *cache*. Consequentemente, reduz-se a taxa de acertos do *cache* (*cache hit*). Os resultados das simulações mostram que há degradação da taxa de acertos do *cache* quanto maior é o número de fluxos de longa duração.

Todos os trabalhos descritos anteriormente que avaliam e/ou propõem contramedidas para os ataques de inundação de interesses e envenenamento de *cache* não consideram a possibilidade de ataques em que consumidores e produtores maliciosos agem em

conluio para gerar, disponibilizar e manipular a popularidade de conteúdos. Avaliar tal ataque, detalhado na Seção 4, é o principal objetivo deste trabalho.

4. Ataque de Negação de Serviço em Conluio Consumidor-Produtor

No ataque de conluio existem pelo menos dois atores: o produtor malicioso e o consumidor malicioso, como ilustrados na Figura 1. O produtor malicioso é responsável por produzir conteúdo malicioso conforme a demanda do consumidor malicioso. Esse conteúdo tem as mesmas características do conteúdo legítimo e, portanto, não terá tratamento diferenciado por nenhum nó CCN. Isso quer dizer que esse tipo de conteúdo ocupará o *cache* dos nós com o mesmo tratamento dos demais conteúdos que trafegam na rede. Os nomes dos conteúdos maliciosos também seguem as especificações da CCN. O consumidor malicioso, por sua vez, solicita conteúdos maliciosos em altas taxas. Na Figura 1, os nós R1 e R2 armazenam conteúdos maliciosos em *cache*, uma vez que estão no caminho entre o consumidor e o produtor malicioso.

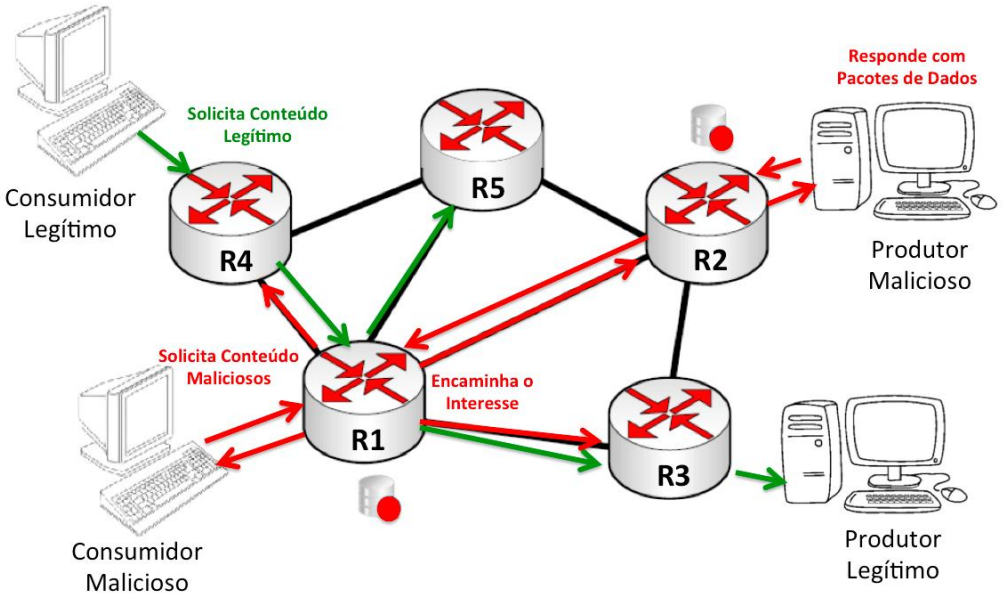


Figura 1. O ataque em conluio consumidor-produtor: nós legítimos e maliciosos em ação.

Com o ataque em conluio, o objetivo é prejudicar o consumo indireto de conteúdos, isto é, obrigar um consumidor legítimo a recuperar o conteúdo desejado diretamente do produtor original. Esse objetivo é alcançado através da manipulação da popularidade dos conteúdos armazenados em *cache*. Consumidores maliciosos enviam pacotes de interesse para um grupo de conteúdos que existem e que são respondidos pelo produtor malicioso. Assim, se solicitado com frequência, um conteúdo se torna popular, apesar de não ter sido solicitado por usuários legítimos. Por isso, o conteúdo é dito malicioso. Esse ataque é possível, porque a CCN emprega políticas de substituição do *cache* baseadas, em sua maioria, na popularidade dos conteúdos. Assim, se um determinado conteúdo não é solicitado com frequência ou não foi solicitado recentemente pelos consumidores, ele é considerado menos popular. Dessa forma, terá prioridade de descarte quando houver necessidade de armazenar novos conteúdos. Ao solicitar um conjunto

específico de conteúdos e em taxas altas, os nós maliciosos manipulam a política de *cache*. Com mais conteúdos maliciosos em *cache*, maior a taxa de erro para os conteúdos legítimos e, consequentemente, a necessidade de nós legítimos terem que recuperar o conteúdo diretamente do seu produtor original. Mesmo que os consumidores legítimos não tenham que consumir diretamente dos produtores originais, eles terão seus interesses encaminhados por mais saltos até conseguir o conteúdo desejado. No exemplo da Figura 1, o consumidor legítimo pode ter que recuperar o conteúdo solicitado diretamente do produtor legítimo, uma vez que o *cache* do nó R1 que está no caminho entre os dois, pode estar sobrecarregado com conteúdo malicioso.

Uma das principais razões para que o ataque em conluio consumidor-produtor seja bem sucedido é o fato de que os pacotes de interesse e de dados usados no ataque são legítimos para a rede e, portanto, não são detectados por mecanismos de verificação de assinaturas. O pacote com o conteúdo malicioso possui uma assinatura válida, carrega a chave do publicador e, assim, passa no teste de verificação de integridade e autenticidade. Logo, não é identificado como malicioso e nem descartado.

Outro objetivo do ataque em conluio é reduzir a eficiência da PIT, ao enviar pedidos de interesses para diferentes conteúdos maliciosos disponibilizados por produtores maliciosos a uma alta taxa. Dessa forma, é possível burlar o mecanismo de *push-back* proposto por Gasti *et al.*. Esse mecanismo é eficiente contra a inundação de pacotes de interesse porque consegue identificar prefixos de nomes de conteúdo que frequentemente estão pendentes na PIT, uma vez que o conteúdo solicitado é inexistente. Porém se o consumidor e produtor estiverem agindo em conluio, os pacotes de interesse terão uma entrada na PIT de um nó somente até o conteúdo malicioso, que existe, retornar. Portanto, o mecanismo *push-back* não terá sucesso ao tentar identificar o ataque, pois os pacotes de interesse receberão uma resposta legítima e suas entradas serão removidas da PIT. Nesse caso, o ataque em conluio não provoca o esgotamento de recursos de armazenamento da PIT em um nó. O objetivo do ataque é gerar uma grande quantidade de pacotes de interesses, fazendo com que um nó tenha que manipular muitas solicitações de conteúdo maliciosas em detrimento a interesses legítimos, o que pode levar a negação de serviço nesse nó.

5. Cenário de Avaliação

A topologia da rede usada na avaliação do impacto do ataque em conluio é composta por 32 nós dispostos em forma de árvore, como mostra a Figura 2. Os 24 nós folha são consumidores. O número de consumidores legítimos (CL) é fixo em todas as configurações e igual a 16. O número de consumidores maliciosos (CA) varia de 0 a 8. A posição dos CLs e dos CAs é definida aleatoriamente em cada rodada de simulação. O produtor legítimo (PL) é sempre o nó raiz. O produtor malicioso (PA) é o nó filho do nó raiz. Os demais 6 nós que compõem a topologia são os roteadores da rede (RTR). Os enlaces que interconectam os nós possuem taxa de transmissão de 100 Mb/s e atraso de 1 ms.

Os conteúdos são solicitados da seguinte forma. Os consumidores maliciosos enviam interesses para 12 conteúdos que são disponibilizados pelo único produtor malicioso a taxas de 10, 100 e 500 interesses por segundo. Cada conteúdo malicioso possui 100 pedaços (*chunks*) e prefixos de nome diferentes. Os pedaços de conteúdo são solicitados

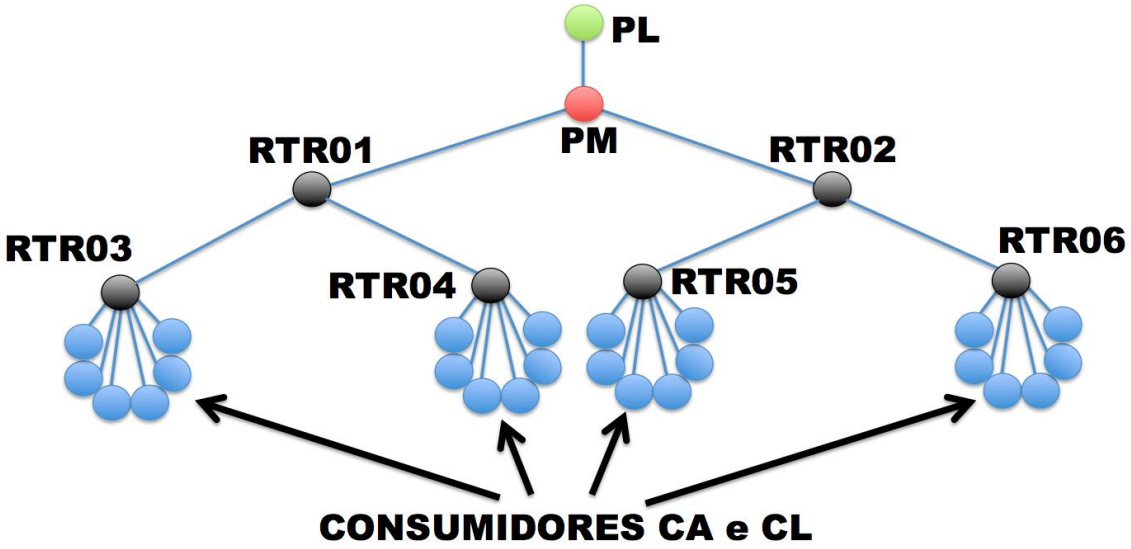


Figura 2. A topologia da rede usada nas simulações.

de duas formas diferentes: o consumo segundo a popularidade do conteúdo malicioso, seguindo uma distribuição *Zipf* com parâmetro $\alpha = 0,7$ [Breslau et al. 1999], e o consumo sequencial, dito CBR, no qual um consumidor envia pacotes de interesse ordenados pelo nome do conteúdo e de forma cíclica. Os consumidores legítimos sempre enviam 10 interesses/s para outros 12 conteúdos disponibilizados pelo produtor legítimo. Cada conteúdo malicioso possui 100 pedaços (*chunks*) e prefixos de nome diferentes. Os pedaços são solicitados seguindo uma distribuição *Zipf* com parâmetro $\alpha = 0,7$.

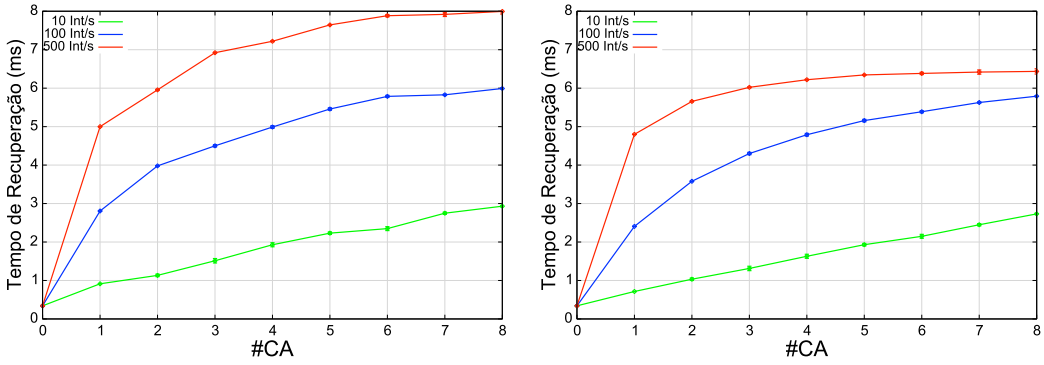
O *cache* dos consumidores legítimos e dos roteadores tem capacidade para armazenar até 1000 pedaços de conteúdo e cada pedaço possui 1024 bytes. Os consumidores maliciosos não possuem *cache* para potencializar o ataque, isto é, sempre enviam interesses independentemente se já receberam o conteúdo anteriormente ou não. A PIT tem tamanho ilimitado para que seja possível avaliar apenas o efeito do aumento da ocupação maliciosa no *cache* dos nós. A política de substituição de *cache* é a *Least Recently Used* (LRU).

O módulo ndnSIM do simulador NS-3 é usado na avaliação. Para cada configuração, são realizadas 50 rodadas de simulação, cada uma com duração de 180 s. Para os pontos dos gráficos obtidos, são calculados intervalos de confiança representados por barras verticais para um nível de confiabilidade de 95%.

6. Resultados

Os resultados apresentados têm como objetivo avaliar o impacto do ataque de negação de serviço em conluio consumidor-produtor no desempenho da CCN. As métricas de desempenho são o tempo médio de recuperação de conteúdos legítimos, a ocupação maliciosa média do *cache* dos roteadores, a taxa média de erros de *cache* dos conteúdos legítimos e o percentual de conteúdos legítimos recuperados do produtor original. A Figura 3 mostra o comportamento do tempo médio de recuperação de conteúdos legítimos em função do número de consumidores maliciosos. Nas duas configurações, quando consumidores maliciosos solicitam conteúdos segundo o padrão

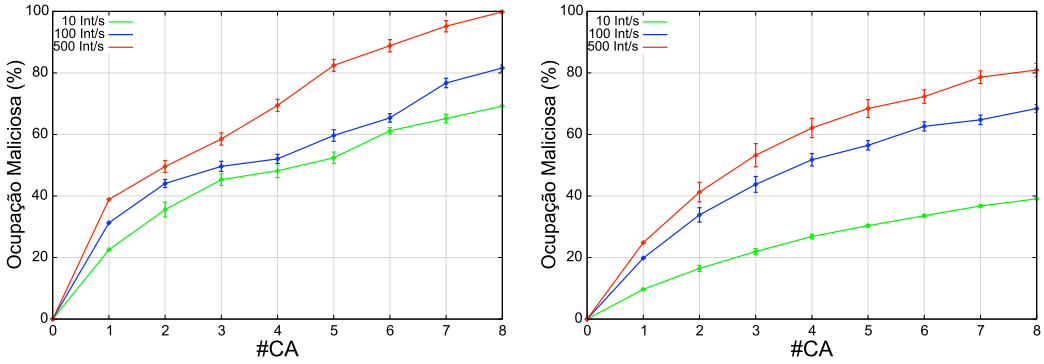
CBR (Figura 3(a)) ou quando solicitam conteúdos segundo a distribuição *Zipf* (Figura 3(b)), o comportamento observado é o mesmo: quanto mais consumidores maliciosos, maior o tempo médio de recuperação de conteúdos. Da mesma forma, quanto maior a taxa de interesses maliciosos, maior o tempo médio de recuperação de conteúdos legítimos. Para a configuração da Figura 3(a), por exemplo, quando somente consumidores legítimos solicitam conteúdos, o tempo médio de recuperação de conteúdos legítimos é igual a 0,34 ms. Por outro lado, quando 4 consumidores maliciosos solicitam conteúdos esse tempo é igual a 1,92 ms e 7,21ms, quando enviam 10 e 500 interesses/s, respectivamente. Quando há 8 consumidores maliciosos, o tempo médio de recuperação de conteúdos legítimos é igual a 2,93 ms e 7,99 ms para as taxas de 10 e 500 interesses/s, respectivamente. Isso mostra que o tempo de recuperação aumentou 23,5 vezes no pior caso para as configurações avaliadas. É importante ressaltar que como os consumidores legítimos possuem *cache* e como eles sempre consomem de acordo com a popularidade (*Zipf*), pode-se observar que sem ataque o consumo é, muitas vezes, feito do próprio *cache* do nó, o que resulta em um tempo de recuperação inferior a 1 ms.



(a) CBR. (b) Zipf.

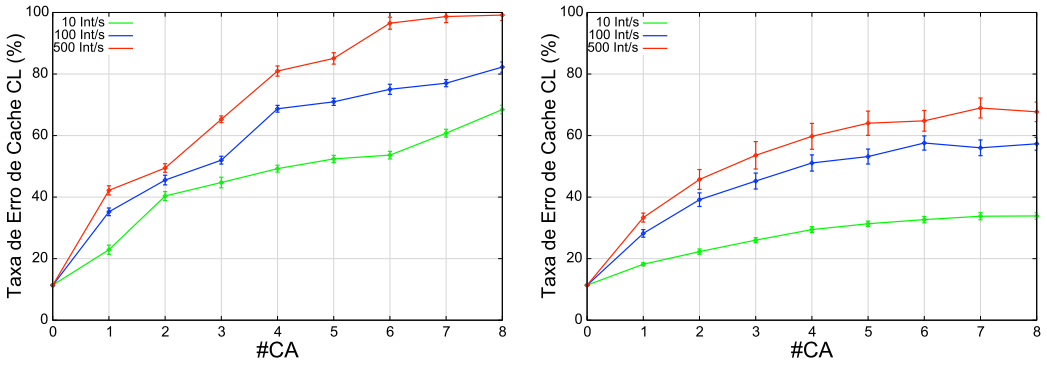
Figura 3. O tempo de recuperação de conteúdos legítimos.

O aumento do tempo de recuperação nas duas configurações é explicado pelo aumento da ocupação maliciosa no *cache* dos nós intermediários e, consequentemente, do aumento da taxa de erros de *cache*, como mostram as Figuras 4 e 5, respectivamente.



(a) CBR. (b) Zipf.

Figura 4. O percentual de ocupação do *cache* dos roteadores por conteúdos maliciosos.



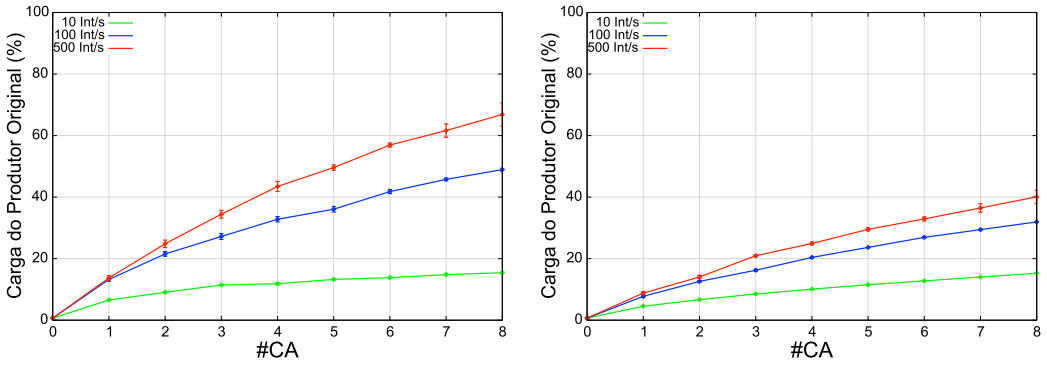
(a) CBR. (b) Zipf.

Figura 5. A taxa de erros de *cache* para os conteúdos legítimos.

Quanto maior a ocupação maliciosa, maior a probabilidade do conteúdo solicitado não estar armazenado em *cache*. Para a configuração da Figura 3(a) e curva para a taxa de 100 interesses/s, por exemplo, nota-se que com 8 consumidores maliciosos, o tempo de recuperação é da ordem de 6 ms. Como o atraso de cada enlace é de 1 ms, conclui-se que os conteúdos legítimos são recuperados mais frequentemente de nós que estão a mais saltos do consumidor do que os nós de borda. Isso indica que os roteadores de borda estão com uma alta ocupação de conteúdos maliciosos em seu *cache*. Nessa situação, os conteúdos maliciosos ocupam em média 80% do espaço total do *cache* dos roteadores, como mostra a Figura 4(a). Para a configuração na qual os consumidores maliciosos solicitam conteúdos com base na sua popularidade (Figura 3(b)), o tempo médio de recuperação também aumenta, mas esse aumento é menor do que o observado para a configuração baseada no consumo CBR (Figura 3(a)). Isso porque os consumidores maliciosos têm maior probabilidade de encontrarem conteúdos solicitados anteriormente em nós que estão a um ou dois saltos de distância e seus interesses não chegam ao produtor malicioso. Assim, a ocupação maliciosa na rede é menor (Figura 4(b)), em especial nos nós próximos ao produtor original. Assim, tem-se uma taxa de erros de *cache* menor (Figura 5(b))e um tempo de recuperação menor para consumidores legítimos.

A Figura 6 mostra o percentual de conteúdos legítimos recuperados do produtor original em função do número de consumidores maliciosos e da taxa de envio de interesses por esses nós. Esses resultados corroboram que o ataque em conluio reduz e eficiência do emprego do *cache* pela CCN. A Figura 6(a) mostra que se não há ataque cerca de 0,5% dos conteúdos solicitados são recuperados diretamente do produtor original. Nesse caso, cada conteúdo legítimo é recuperado do produtor original no máximo duas vezes, até que seja armazenado pelos nós RTR1 e RTR2 (Figura 2). Porém, basta se ter 4 consumidores maliciosos operando a taxa de 10 interesses/s para que esse valor aumente para cerca de 12%. No pior caso, os consumidores legítimos estão recuperando cerca de 67% dos conteúdos legítimos diretamente do produtor original.

Um resultado interessante é que uma ocupação maliciosa de quase 100% quando há 8 consumidores maliciosos enviando 500 interesses/s (Figura 4(a)) não resulta em 100% de conteúdos recuperados do produtor original (Figura 6(a)). Tal fato é explicado pelo uso de *cache* pelos próprios consumidores. Assim, é possível recuperar o conteúdo do próprio *cache*, sem ter que encaminhar pacotes de interesse para outros nós. No en-



(a) CBR. (b) Zipf.

Figura 6. Percentual de carga do produtor original.

tanto, quando sob ataque, os consumidores legítimos ainda têm solicitações de conteúdo encaminhadas até o produtor original, mesmo que eles façam uso de *cache* e solicitem conteúdos de acordo com a popularidade. Portanto, isso comprova que o serviço é negado em virtude da ocupação maliciosa dos *caches* dos roteadores.

Outra observação interessante extraída dos resultados é que a distribuição de consumidores maliciosos é mais efetiva do que o aumento da taxa agregada de envio de interesses maliciosos. Por exemplo, na Figura 3(a), é possível observar que o tempo médio de recuperação de conteúdos legítimos é da ordem de 5 ms quando 4 consumidores maliciosos enviam 100 interesses/s cada um (taxa agregada de 400 interesses/s) ou quando um consumidor malicioso envia sozinho 500 interesses/s. Esse fato é explicado pela ocupação maliciosa dos *caches* ser mais efetiva quando o ataque é distribuído. Para o mesmo exemplo anterior, a Figura 4(a) mostra que a ocupação maliciosa quando há 4 atacantes enviando 100 interesses/s é da ordem de 50%. Quando há um atacante apenas enviando 500 interesses/s ela é de 40%. Esse fato se repete em outros pontos dos gráficos, considerando também o consumo baseado na popularidade. Por exemplo, na Figura 3(b), quando 8 consumidores maliciosos enviam 10 interesses/s cada um (taxa agregada de 80 interesses/s) o tempo médio de recuperação de conteúdos legítimos é da ordem de 3 ms. Se um consumidor malicioso envia sozinho 100 interesses/s, esse tempo é menor do que 2,5 ms.

É importante ressaltar que em todos os experimentos realizados, os consumidores legítimos recuperaram todos os conteúdos solicitados. Isso pode ser explicado pelo fato da PIT não ter seu tamanho limitado e por ter sido usado um temporizador para remoção de entradas desta tabela da ordem de 4 segundos. Nas configurações usada, esse tempo é muito maior do que o tempo necessário para um pacote de interesse legítimo ser encaminhado até o produtor original e o pacote de dados ser encaminhado pelo caminho reverso até o consumidor. No pior caso, como mostra a Figura 3(a), esse tempo é de aproximadamente 8 ms. Além disso, o produtor original nunca remove do seu *cache* o conteúdo produzido por ele próprio. Portanto, nenhum dos nós da rede remove uma entrada da PIT nas configurações usadas antes do consumidor legítimo receber o conteúdo solicitado, mesmo que para isso, os pacotes de interesse tenham que ser encaminhados até o produtor original.

7. Conclusão

Este trabalho avaliou o ataque de negação de serviço em conluio consumidor-produtor para a arquitetura CCN. Esse ataque visa aumentar o tempo de recuperação de conteúdos aumentando a ocupação do *cache* dos nós intermediários com conteúdos maliciosos. Além disso, o ataque em conluio consumidor-produtor, não é identificado pelo mecanismo padrão de verificação de assinaturas da CCN porque os pacotes maliciosos carregam uma assinatura digital válida.

Diferentes configurações foram usadas nas simulações, variando-se o número de consumidores maliciosos, a política de consumo desses consumidores e taxa de pacotes de interesse maliciosos. Os resultados mostram que o ataque em conluio é efetivo, o que compromete o emprego do *cache* pela CCN. No pior caso, o tempo de recuperação aumentou 23,5 vezes para as configurações avaliadas. Esse aumento se deve a uma ocupação maliciosa média de 99% e, consequentemente, a uma taxa de erro de *cache* de 99%. Com isso, os consumidores legítimos recuperando 67% dos conteúdos solicitados diretamente do produtor original. Mostra-se também que a distribuição de consumidores maliciosos é mais efetiva do que o aumento da taxa agregada de envio de interesses maliciosos. Os trabalhos futuros incluem a avaliação de outras políticas de substituição de *cache* e do emprego do mecanismo de *push-back* proposto por Gasti *et al.*. O objetivo é verificar se tal mecanismo é eficiente como contramedida ao ataque em conluio. Caso não seja, o próximo passo é propor uma contramedida. Sobre os cenários, pretende-se empregar topologias reais e com maior escala nas simulações.

Agradecimentos

Este trabalho é apoiado por Dinter UFF/UFAC, CNPq, CAPES, FAPERJ, Proppi/UFF, TBE/ANEEL e CELESC/ANEEL.

Referências

Afanasyev, A., Mahadevan, P., Moiseenko, I., Uzun, E. e Zhang, L. (2013). Interest flooding attack and countermeasures in named data networking. Em *IFIP Networking*, páginas 1–9.

Breslau, L., Cao, P., Fan, L., Phillips, G. e Shenker, S. (1999). Web caching and zipf-like distributions: Evidence and implications. Em *IEEE Conference on Computer Communications - INFOCOM*, páginas 126–134.

Brito, G. M., Velloso, P. B. e Moraes, I. M. (2012). Redes orientadas a conteúdo: Um novo paradigma para a Internet. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC*, páginas 211–264.

Brito, G. M., Velloso, P. B. e Moraes, I. M. (2013). *Information-Centric Networks, A New Paradigm for the Internet*. FOCUS - Networks and Telecommunications Series. Wiley-ISTE, 1 edition.

Choi, S., Kim, K., Kim, S. e Roh, B. (2013). Threat of DoS by interest flooding attack in content-centric networking. Em *Information Networking International Conference*, páginas 315–319.

Gasti, P., Tsudik, G., Uzun, E. e Zhang, L. (2013). DoS and DDoS in named-data networking. Em *International Conference on Computer Communications and Networks - ICCCN*, páginas 1–7.

Guimarães, F. Q., Ribeiro, I. C. G., de A. Rocha, A. A. e Albuquerque, C. V. N. (2013). Nem tanto, nem tão pouco: Existe um timeout Ótimo para PIT CCN na mitigação de ataques DoS. Em *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)*, páginas 253–266.

Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N. e Braynard, R. (2009). Networking named content. Em *International Conference on emerging Networking EXperiments and Technologies - CoNEXT*, páginas 1–12.

Kim, Y., Kim, U. e Yeoml, I. (2013). The impact of large flows in content centric networks. Em *IEEE International Conference on Network Protocols - ICNP*, páginas 1–2.

Ribeiro, I. C. G., de A. Rocha, A. A., Albuquerque, C. V. N. e Guimarães, F. Q. (2014). On the possibility of mitigating content pollution in content-centric networking. Em *Conference on Local Computer Networks (LCN)*, páginas 498–501.

Ribeiro, I. C. G., de Q. Guimarães, F., Albuquerque, C. V. N. e de A. Rocha, A. A. (2013). CCNcheck: um mecanismo de mitigação para poluição de conteúdos em redes centradas em conteúdo. Em *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)*, páginas 114–127.

Ribeiro, I. C. G., Guimarães, F. Q., Kazienko, J. F., de A. Rocha, A. A., Velloso, P. B., Moraes, I. M. e Albuquerque, C. V. N. (2012). Segurança em redes centradas em conteúdo: Vulnerabilidades, ataques e contramedidas. Em *Minicursos do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg*, páginas 101–150.

Smetters, D. e Jacobson, V. (2009). Securing network content. Relatório Técnico TR-2009-1, Xerox Palo Alto Research Center - PARC.

Xie, M., Widjaja, I. e Wang, H. (2012). Enhancing cache robustness for content-centric networking. Em *IEEE Conference on Computer Communications - INFOCOM*, páginas 2426–2434.

Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornton, J., Smetters, D. K., Zhang, B., Tsudik, G., Claffy, K., Krioukov, D., Massey, D., Papadopoulos, C., Abdelzaher, T., Wang, L., Crowley, P. e Yeh, E. (2010). Named Data Networking (NDN) project. Relatório Técnico NDN-0001, Xerox Palo Alto Research Center - PARC.

Estimativa e Sinalização de Congestionamentos de Tráfego através de Redes Veiculares V2V

Roniel Soares de Sousa[1], **André Castelo Branco Soares**[1]

[1]Programa de Pós-graduação em Ciência da Computação (PPGCC)
Universidade Federal do Piauí (UFPI)
Caixa Postal 64.049-550 – Teresina – PI – Brasil

ronizsds@gmail.com, andre.soares@ufpi.edu.br

Abstract. *The increase in traffic congestion in big cities causes financial losses annually. Intelligent Transportation Systems are used in various ways to improve the urban mobility (dynamic adjustment of traffic light timers, traffic monitoring, etc). This paper proposes a protocol to estimate and signal collaboratively congestion levels using the V2V (Vehicle-to-Vehicle) communication. The protocol was developed to efficiently use the wireless channel, reducing the signaling overhead. Through simulations, we verified the proposed protocol ability to detect and disseminate congestion information. The proposed protocol was compared to a solution present in the literature and our protocol outperformed in terms of signaling overhead.*

Resumo. *O aumento dos congestionamentos de trânsito em grandes cidades causam prejuízos financeiros anualmente. Sistemas Inteligentes de Transporte são utilizados de diversas formas para melhorar a mobilidade urbana (ajuste dinâmico nos temporizadores de semáforos, monitoramento de tráfego, etc). Este trabalho propõe um protocolo para estimar e sinalizar de forma colaborativa os níveis de congestionamento usando comunicação V2V (Vehicle-to-Vehicle). O protocolo foi desenvolvido para utilizar eficientemente o meio sem fio, reduzindo o overhead de sinalização. Através de simulações, verificou-se a capacidade do protocolo proposto em detectar e divulgar informações de congestionamentos. O protocolo proposto foi comparado com uma solução presente na literatura e obteve melhor desempenho em termos de overhead de sinalização.*

1. Introdução

O constante aumento dos congestionamentos de trânsito em grandes cidades causam impactos negativos em diversas áreas, como na economia, no meio ambiente, na saúde, dentre outras. Segundo a Fundação Getúlio Vargas, os congestionamentos causaram prejuízos de cerca de R$ 40 bilhões na cidade de São Paulo, somente no ano de 2012 [Cintra 2013]. O uso de Sistemas Inteligentes de Transportes é uma das principais formas de atacar este problema [Aissaoui et al. 2014]. Recentemente, diversas soluções para monitoramento de tráfego baseadas na comunicação entre veículos através do padrão 802.11p vem sendo propostas [Araújo et al. 2014], [Gramaglia et al. 2014], [Silva et al. 2013], [Wischoff et al. 2003], [Younes and Boukerche 2013], [Aissaoui et al. 2014], [Milojevic and Rakocevic 2014].

As Redes Veiculares, conhecidas como VANETs (*Vehicular Ad Hoc Networks*), são redes que permitem a troca de informações entre veículos (comunicação V2V - *Vehicle-to-Vehicle*) e entre veículos e infra-estruturas localizadas às margens das vias (comunicação V2I - *Vehicle-to-Infrastructure*). As VANETs possuem características particulares que representam desafios na comunicação entre veículos. Algumas dessas características são: (i) a alta mobilidade dos nós, (ii) mudanças constantes na topologia da rede, (iii) a existência de redes densas que ocasionam altas taxas de perdas de pacotes por congestionamento na rede e (iv) a existência de redes esparsas que dificultam a disseminação de informações [dos S. Alves et al. 2009].

As aplicações de Redes Veiculares podem ser divididas em três classes: segurança no trânsito, entretenimento e assistência ao motorista. O principal desafio para as aplicações de segurança é divulgar rapidamente as informações para que o condutor tenha tempo para reagir. Nas aplicações de entretenimento destacam-se os sistemas de compartilhamento de conteúdo (músicas, filmes) e jogos. Já as aplicações de assistência ao motorista são aquelas que auxiliam o condutor, como por exemplo através de informações sobre as condições de trânsito [dos S. Alves et al. 2009].

Este trabalho tem foco no problema do monitoramento e da engenharia de tráfego, visando a redução de congestionamento de veículos. Este artigo propõem um protocolo para estimativa e sinalização de congestionamentos de tráfego em redes veiculares com baixo *overhead*. O desempenho do protocolo proposto é comparado com a solução apresentada em [Milojevic and Rakocevic 2014]. De maneira geral, o algoritmo proposto reduziu o tempo de viagem dos veículos. Além disso, quando comparado com a solução proposta em [Milojevic and Rakocevic 2014], o protocolo proposto reduziu em mais de 93% o *overhead* com sinalização.

O restante deste artigo está organizado como segue. A Seção 2 apresenta os principais trabalhos relacionados. A Seção 3 detalha o funcionamento do trabalho proposto. As simulações e os resultados do estudo de avaliação de desempenho do protocolo são apresentados na Seção 4. Por fim, a Seção 5 apresenta as conclusões e os potenciais trabalhos futuros.

2. Trabalhos Relacionados

Diversas soluções para controle de congestionamentos veiculares baseadas na comunicação entre veículos vem sendo propostas [Wischoff et al. 2003] [Younes and Boukerche 2013] [Silva et al. 2013] [Gramaglia et al. 2014] [Araújo et al. 2014] [Milojevic and Rakocevic 2014]. De maneira geral, tais soluções têm algumas das seguintes características: estimativa dos níveis de congestionamento, divulgação de informações das condições de tráfego, minimização, predição e prevenção de congestionamentos.

Em [Wischoff et al. 2003] é apresentado um Sistema de Informações de Tráfego chamado SOTIS. No SOTIS cada veículo é capaz de analisar as condições de tráfego por meio de mensagens enviadas por outros veículos. Essas mensagens possuem informações de velocidade, posição e identificação de qual via cada veículo se encontra. As mensagens são enviadas periodicamente em *broadcasts*. Os autores em [Wischoff et al. 2003] avaliaram o SOTIS através de simulações em um cenário de avenidas. Nas simulações o único fator estudado foi o atraso no envio das mensagens.

Outras soluções para detecção de congestionamentos necessitam obrigatoriamente da comunicação V2I, como os trabalhos apresentados em [Silva et al. 2013] e [Younes and Boukerche 2013]. Em [Silva et al. 2013] é implementado um *middleware* chamado ConProVa que depende exclusivamente da comunicação V2I. Já o trabalho de [Younes and Boukerche 2013], chamado de ECODE, utiliza tanto a comunicação V2V como a comunicação V2I.

Em [Gramaglia et al. 2014] é proposta uma solução para monitoramento de tráfego chamada ABEONA. O ABEONA permite o monitoramento de tráfego em tempo real de uma maneira flexível e com um custo eficiente. ABEONA faz uso do conhecimento cooperativo e distribuído das condições de tráfego (velocidade, fluxo, densidade veicular média) e é capaz de prever as condições de tráfego em uma janela de 15 a 20 minutos. Os autores em [Gramaglia et al. 2014] não tratam de como as informações podem ser disseminadas e nem avaliam o impacto do uso de tais informações para melhorar as condições de tráfego das vias.

Um protocolo para detecção e redução de congestionamento em redes veiculares chamado CARTIM é proposto em [Araújo et al. 2014]. A proposta utiliza um sistema baseado em lógica *fuzzy* que possui como variáveis de entrada a velocidade do veículo e a densidade veicular da via. Através dessas informações o nível de congestionamento local é estimado. A validação das estimativas de níveis de congestionamento são feitas com a troca de mensagens entre os veículos que estão presentes na via. Esse processo de validação é realizado com encaminhamento de mensagens *multi-hop* e *unicast*, partindo da extremidade dianteira do congestionamento para a extremidade traseira. Ainda em [Araújo et al. 2014] os autores propõem uma heurística para minimização dos congestionamentos de tráfego. Assim que o nível de congestionamento atingir um determinado limiar, os veículos passam a transmitir mensagens CTA (*Congestion Traffic Avoid*) sinalizado para outros veículos que eles devem evitar passar na via sob condições severas. Para que as mensagens CTA sejam transmitidas para veículos em outras vias, é utilizado também a comunicação V2I. Tal abordagem impede o uso da referida proposta em cenários sem unidades de acostamento.

Em [Milojevic and Rakocevic 2014], os autores propõem um algoritmo distribuído para quantificação de congestionamentos de trânsito. Através desse algoritmo, cada veículo é capaz de estimar o nível de congestionamento local baseando-se apenas na velocidade atual. Através da troca de mensagens com outros veículos é possível validar estas informações e obter informações de outras vias. O algoritmo foi feito buscando atingir os seguintes objetivos: eficiência em termos de mensagens trocadas, completamente distribuído e independente de qualquer tipo de infraestrutura ou sistemas externos. Nesse trabalho, os veículos precisam obter informações sobre os níveis de congestionamentos de outras vias, mesmo aquelas nas quais o veículo não irá passar. Os autores em [Milojevic and Rakocevic 2014] não avaliam o impacto da sinalização sobre o congestionamento na eficiência do deslocamento dos veículos, o que deveria reduzir os níveis de congestionamentos ou evitar que os veículos alcancem áreas congestionadas. O trabalho desenvolvido em [Milojevic and Rakocevic 2014] limita-se a apenas detectar os congestionamentos.

O protocolo proposto neste artigo busca superar algumas das limitações dos trabalhos relacionados nesta seção. Assim, a contribuição deste artigo está na proposta de um

protocolo para minimizar os congestionamentos de veículos sem a exigência de Unidades de Acostamento. Em especial, destaca-se que o protocolo proposto usa de forma mais eficiente o meio sem fio, reduzindo o *overhead* de sinalização.

3. Protocolo de Sinalização de Congestionamento com Baixo *Overhead*

O protocolo proposto neste trabalho para sinalização de congestionamento com baixo overhead é apresentado nesta seção. O protocolo proposto faz uso de três mecanismos: (i) detecção e quantificação de congestionamento, (ii) disseminação de informação e (iii) minimização de congestionamento. Ressalta-se que o mecanismo de detecção e quantificação de congestionamento utilizado é baseado no protocolo apresentado em [Milojevic and Rakocevic 2014].

3.1. Detecção e Quantificação de Congestionamento

Uma das definições de congestionamento de trânsito mais utilizadas na literatura é a apresentada em [Lomax et al. 1997]. Os autores em [Lomax et al. 1997] definem o congestionamento de trânsito como um tempo de viagem ou atraso superior ao que é necessário para realizar o mesmo deslocamento em condições de baixo fluxo de veículos. Em [Rao and Rao 2012] os autores apresentam diferentes métricas para identificar e medir o congestionamento veicular urbano. Os autores concluem que congestionamento é uma função da redução nas velocidades dos veículos. Os autores em [Rao and Rao 2012] também apontam a utilidade de medir o congestionamento relacionando a velocidade dos veículos a um limiar de velocidade apropriado para uma dada via de trânsito.

Entrada:
T: Tempo desde a última vez que a velocidade instantânea do veículo ultrapassou o limiar V_t
V_c: Velocidade atual do veículo
V_t: Limiar de velocidade de referência
C_L: Nível de congestionamento atual.
Saída: Novo valor para C_L
inicio
  se $V_c > V_t$ *e* $T > 10s$ **então**
    retorne 1;
  fim
  se $V_c \leq V_t$ **então**
    retorne $max(10, \lfloor \frac{T+19}{20} \rfloor)$;
  fim
  retorne C_L;
fin

Algoritmo 1: Algoritmo que altera o valor de C_L.

De acordo com [Rao and Rao 2012], neste contexto a velocidade atual de cada veículo (V_c) é utilizada como parâmetro para quantificar o nível de congestionamento. Assume-se que cada via de trânsito tem um limiar de velocidade (V_t) de referência para quantificar o nível de congestionamento da própria via. Assim como em [Milojevic and Rakocevic 2014], neste artigo adotou-se $V_t = 6m/s$. Dessa forma, cada

veículo calcula um nível de congestionamento local C_L, analisando o período de tempo em que a sua velocidade se mantem acima ou abaixo de V_t. C_L pode assumir valores discretos de 1 a 10. O valor inicial de C_L é 1, o que representa fluxo livre. Os valores de 2 a 10 são reservados para situações onde a velocidade é menor que o limiar V_t. O Algoritmo 1 é utilizado para atualizar o valor de C_L. A função *max* retorna o maior valor dentre os passados como parâmetros.

A variável T refere-se ao período de tempo em que a velocidade do veículo (V_c) se mantem acima ou abaixo do V_t. De acordo com as equações, a cada 20 segundos, o valor de C_L é incrementado em 1 unidade se V_c se mantiver abaixo do limiar. Se V_c for mantida abaixo de V_t por mais de 200 segundos o congestionamento será máximo ($C_L = 10$). Se V_c for mantida acima de V_t por 10 segundos o valor de C_L é igual a 1, independente do valor anterior de V_t.

A Figura 1 exemplifica a estimativa de congestionamento de um veículo calculada a partir de sua velocidade instantânea V_c. O gráfico da Figura 1 foi gerado através de simulação considerando os aspectos adotados neste trabalho. *Nível de Congestionamento* e *Velocidade* representam as curvas da estimativa de congestionamento e da velocidade instantânea do veículo, respectivamente.

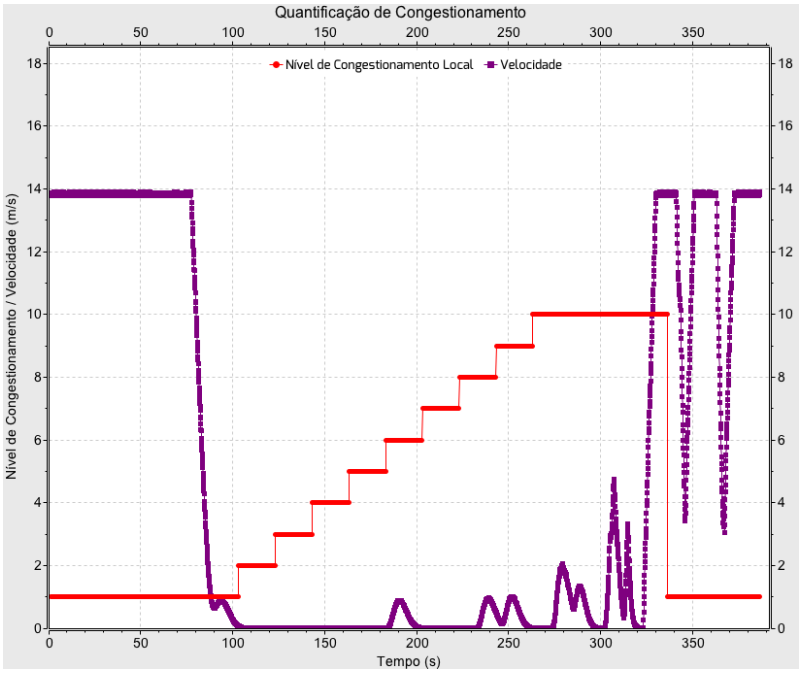


Figura 1. Resultados da quantificação do congestionamento de um veículo e sua velocidade.

Ainda na Figura 1 observa-se as curvas de velocidade e estimativa de congestionamento instantâneos. Durante os 80 segundos iniciais a velocidade do veículo analisado se mantém acima de V_t. Dessa forma, estima-se que a via tem fluxo livre, $C_L = 1$. A partir dos 80 segundos, V_c passa a diminuir com o tempo. Observa-se que, a cada 20 segundos, C_L é incrementado em 1 unidade porque a velocidade do veículo se mantém abaixo de V_t. Em 260 segundos o nível de congestionamento é máximo, $C_L = 10$. A partir dos 340 segundos, V_c aumenta rapidamente, o que significa que o veículo saiu de

um trecho congestionado da via. Note que após 10 segundos, com $V_c > V_t$, a estimativa do congestionamento vai para o valor mínimo, $C_L = 1$.

3.2. Disseminação de Informação

Um dos objetivos do monitoramento de tráfego é proporcionar que os veículos cientes das condições de trânsito utilizem um trajeto menos congestionado para deslocamento. Assim, é importante fazer uso de um mecanismo para o compartilhamento das estimativas de congestionamento entre os veículos. O compartilhamento de informações também é útil para a validação das informações coletadas.

Nesta proposta os veículos transmitem suas estimativas de congestionamento para seus vizinhos através de *beacons*. São transmitidas as estimativas de congestionamento calculadas da via atual e da via percorrida anteriormente. Assim, as seguintes informações são inseridas nos *beacons*: (i) identificador da via atual, (ii) identificador da via percorrida anteriormente, (iii) estimativa de congestionamento da via atual e (iv) estimativa de congestionamento da via percorrida anteriormente. Esse conjunto de informações para um dado veículo é chamado neste artigo de Estimativa Local de Congestionamento (ELC).

Vale ressaltar que as informações dos *beacons* desta proposta podem ser inseridos nas *Cooperative Awareness Messages* (CAMs). As CAMs são definidas pelo *European Telecommunications Standards Institute* (ETSI) para a disseminação de dados de interesse locais. Assim a transmissão dos dados dos *beacons* não elevaria o *overhead* de sinalização. Esta abordagem também é utilizada em outros trabalhos, como em [Gramaglia et al. 2014].

A disseminação das estimativas de congestionamento são transmitidas com o uso de *clusters*. Esta abordagem também é utilizada em [Aissaoui et al. 2014]. A Figura 2 ilustra o processo de escolha do *cluster-head*.

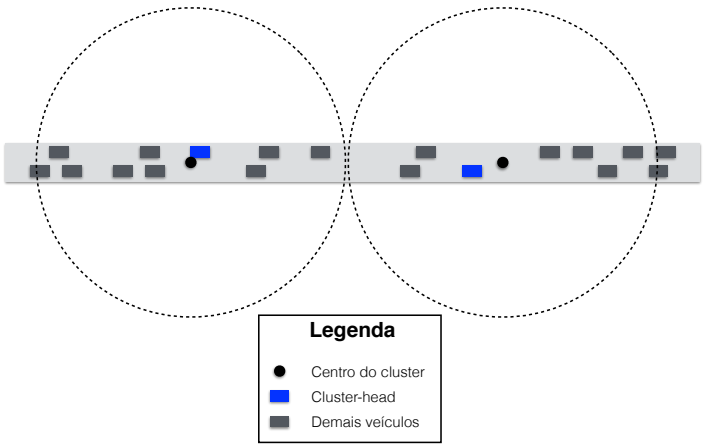


Figura 2. Escolha do cluster-head para disseminação do ELC.

Assume-se que cada veículo possui um mapa digital com uma lista dos centros de todos os *clusters* da cidade. Cada veículo mantém as informações de posições de

seus vizinhos. Estas informações são coletadas dos *beacons* detectados nos últimos 5 segundos [Gramaglia et al. 2014]. Para cada *cluster* um único veículo (*cluster-head*) é escolhido para transmitir em *broadcast* as informações coletadas de seus vizinhos. Na Figura 2 o veículo azul é escolhido para ser o *cluster-head* por estar mais próximo do centro do *cluster*. O *cluster-head* agrega as informações dos veículos pertencentes ao seu *cluster*. As estimativas de congestionamento enviadas pelo *cluster-head* são transmitidas em multi-hop para alcançar os veículos fora do *cluster* emissor.

Cada veículo mantém uma tabela com as estimativas de congestionamentos das diversas vias da região. Essa tabela é preenchida com as estimativas de congestionamento do próprio veículo e com as estimativas recebidas de outros veículos. Destaca-se que cada veículo recebe estimativas de congestionamento dos seus vizinhos (através dos *beacons*) e de veículos de outros clusters. A tabela com as estimativas de congestionamentos é atualizada sempre que um veículo recebe novas estimativas. A atualização da tabela segue uma média simples entre a estimativa atual e a nova estimativa recebida.

Conforme mencionado anteriormente, cada veículo calcula uma nova estimativa de congestionamento com a mudança de sua velocidade. Se essa nova estimativa for maior ou igual a estimativa armazenada em sua tabela (de sua própria via), os seguintes passos são realizados para decidir sobre o envio da mensagem para os demais veículos com as novas estimativas. Primeiro verifica-se a qual cluster o veículo em questão pertence. Para isso verifica-se qual é o *cluster* com o centro mais próximo da posição atual do veículo. A mensagem somente será enviada se o veículo for o mais próximo do centro do seu *cluster* (neste caso o veículo em questão é o *cluster-head*). Vale lembrar que o posicionamento dos vizinhos é identificado com a leitura dos seus *beacons*. A mensagem enviada pelo *cluster-head* contem o ELC do próprio *cluster-head* e o ELC de seus vizinhos. Para isso, cada veículo reune os $ELCs$ de até 10 vizinhos recebidos nos últimos 5 segundos.

Cada veículo mantém uma tabela com informações de estimativas de congestionamento de diversas vias. Essa tabela é montada através das mensagens recebidas de outros veículos. A Figura 3 apresenta uma comparação entre a estimativa local de congestionamento (*Nível de Congestionamento Local*) de um determinado veículo e o valor salvo na tabela (*Nível de Congestionamento no Banco de Dados*), para uma mesma via. Observa-se que mesmo antes do veículo estimar um aumento no nível de congestionamento (após os 100 segundos), as informações presentes na tabela já indicam um aumento no nível de congestionamento. É importante notar que as estimativas do veículo e as estimativas presentes na tabela se mantém compatíveis no decorrer da simulação.

3.3. Minimização de Congestionamento

Com o objetivo de tentar reduzir o tempo gasto no trânsito pelos motoristas, foi proposta a utilização da tabela que cada veículo possui com as estimativas de congestionamentos das diversas vias da região para o cálculo das rotas. Sempre que uma alteração na tabela for feita o veículo poderá recalcular a sua rota da seguinte forma. Será atribuido um peso para cada via através da Equação 1, onde P é o peso atribuído para a via, T é o tamanho da via em metros, V_{id} é a velocidade máxima permitida para a via e C_L é o nível de congestionamento da via salvo na tabela.

$$P = \frac{T}{V_{id}} \times C_L \tag{1}$$

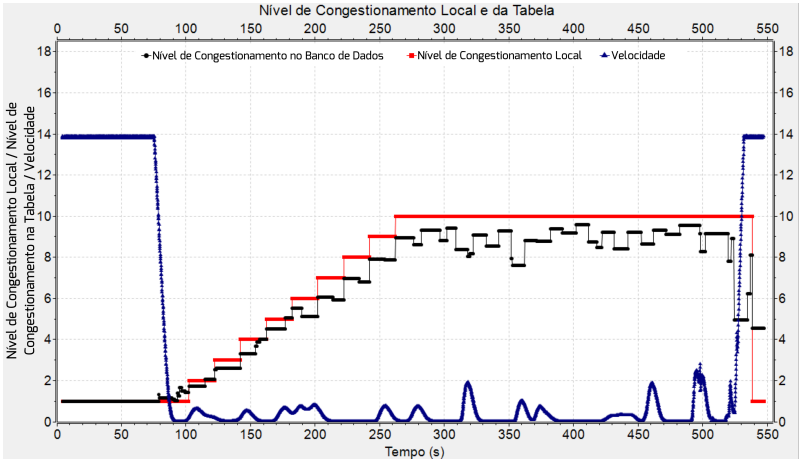


Figura 3. Nível de congestionamento que o veículo calculou localmente através de sua velocidade e o nível de congestionamento recebido de outros veículos para a via atual.

Calculado os pesos de todas as vias, o veículo poderá recalcular sua rota através do algoritmo de menor caminho de Dijkstra.

4. Simulações e Resultados

Simulações foram conduzidas para avaliar o protocolo proposto. A finalidade é atestar a eficácia do protocolo em estimar e sinalizar os congestionamentos de tráfego além de comparar o *overhead* de sinalização com o de [Milojevic and Rakocevic 2014]. Também foram realizadas simulações para avaliar o impacto da proposta na minimização de congestionamentos em termos do tempo médio de viagem dos veículos.

Para este propósito foi utilizada a ferramenta de simulação de rede OMNET++ 4.5 [Varga and Hornig 2008] em conjunto com o simulador de tráfego e mobilidade urbana SUMO 0.21.0 [Krajzewicz et al. 2012]. O *framework* Veins 3.0 [Sommer et al. 2011], que é amplamente utilizado no estudo de redes veiculares [Milojevic and Rakocevic 2014], [Araújo et al. 2014], [Gramaglia et al. 2014] e possui suporte para o padrão 802.11p, foi utilizado para integrar estes dois simuladores.

A potência de transmissão para a comunicação V2V foi de 1.3mW, o que resulta em um raio de transmissão de aproximadamente de 125 metros. Já a frequência de transmissão de *beacons* é de 1 Hz. Foram utilizados os valores padrões para o restante dos parâmetros de simulação.

O cenário de avaliação gerado através do SUMO foi um Manhattan Grid (5x5) sob uma região de 1 Km^2. A Figura 4 apresenta o mapa do cenário de avaliação. Cada segmento de via possui extensão de 250 metros, duas faixas em cada direção e limite de velocidade de 50 km/h. Esse tipo de mapa é comumente utilizado em redes veiculares para avaliar protocolos em ambientes urbanos.

Foram geradas 3600 rotas aleatórias de tamanho mínimo de 1 km. Estas rotas foram criadas escolhendo pares aleatórios de vias origem e destino. Para cada par uma rota era criada composta pela sequência de vias com o menor caminho que ligassem esse par. A velocidade máxima de cada veículo foi configurada para 50km/h. A duração de

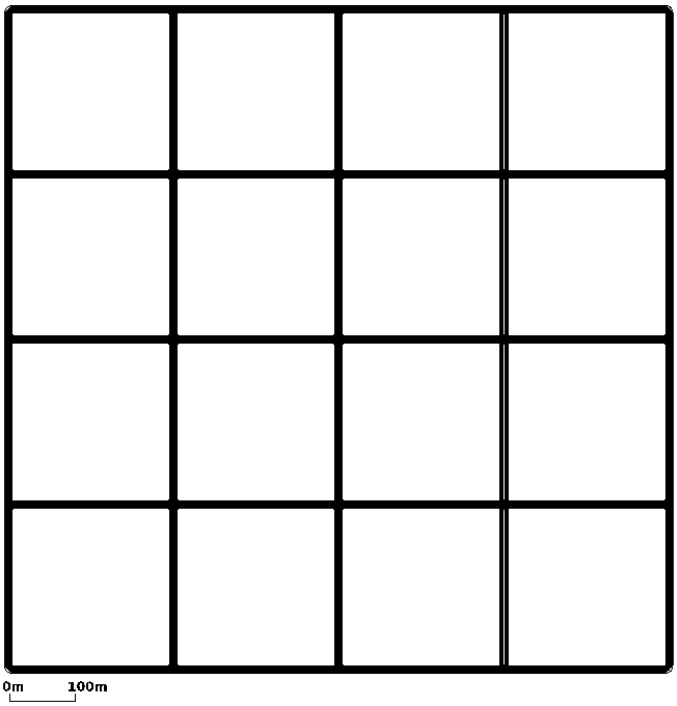


Figura 4. Mapa do cenário de simulação.

cada simulação foi de 1000 segundos. A quantidade de veículos foi mantida constante em 150, 200, 250 e 300, e cada cenário foi replicado 5 vezes, totalizando 20 replicações por cenário. Com o objetivo de avaliar o impacto da proposta na minimização de congestionamentos em termos do tempo médio de viagem dos veículos, o protocolo proposto foi avaliado em dois cenários. No primeiro cenário cada veículo deveria percorrer uma das rotas geradas anteriormente. No segundo cenário também era escolhido uma das rotas geradas anteriormente para cada veículo, porém os veículos poderiam utilizar as estimativas de congestionamento parar alterar as suas rotas, objetivando alcançar a ultima via da rota em menor tempo. No restante deste artigo o primeiro cenário será chamado de **cenário sem alteração de rotas** e o segundo cenário de **cenário com alteração de rotas**.

O desempenho do protocolo proposto foi analisado em cenários com a proposta de minimização de congestionamento (alteração de rotas) e em cenários onde a proposta não foi utilizada. Todos os gráficos apresentados nesta seção possuem intervalos de confiança de 95%.

O primeiro fator avaliado foi o ***overhead* de sinalização**. Nas Figuras 5 e 6 o desempenho do protocolo proposto (com e sem as alterações de rotas) é comparado com o protocolo apresentado em [Milojevic and Rakocevic 2014]. Vale notar que esses valores representam apenas a quantidade de mensagens geradas em cada veículo, excluindo os *beacons* e os encaminhamentos realizados pelo *broadcast multi-hop*. A Figura 5 apresenta a quantidade total de pacotes gerados em cada cenário. Já a Figura 6 apresenta a quantidade média de pacotes gerados por veículo em cada cenário. Pode-se observar que a quantidade de mensagens geradas pela proposta chega a ser cerca 93% menor que a quantidade gerada pelo protocolo de [Milojevic and Rakocevic 2014]. Isso representa uma diferença significativa. A principal caracteristica da proposta que causa este resultado satisfatório

é a utilização de *clusters* para o envio das informações para outras regiões. No cenário com alterações de rotas os níveis de congestionamentos são menores e mais dinâmicos, o que justifica um *overhead* ligeiramente maior quando comparado com o cenário sem alteração de rotas.

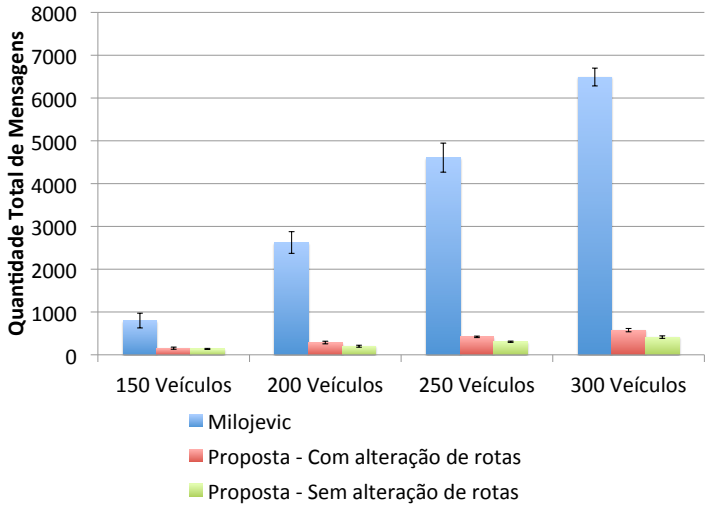


Figura 5. Quantidade total de mensagens geradas para sinalizar as estimativas de congestionamento.

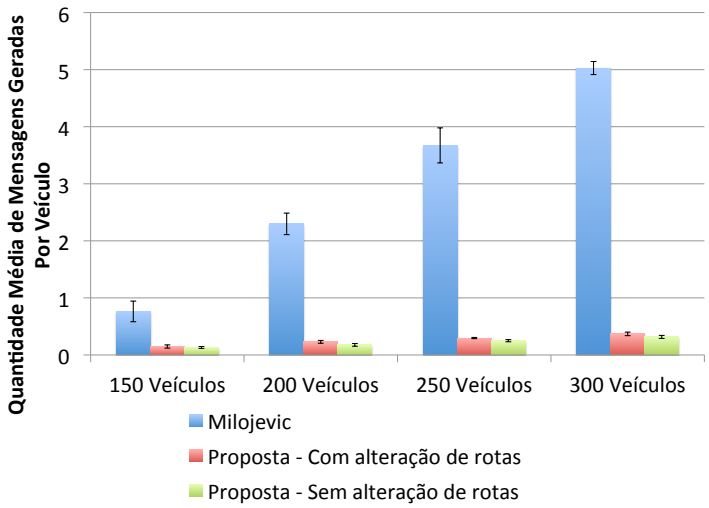


Figura 6. Quantidade média de mensagens geradas por veículo para sinalizar as estimativas de congestionamento.

O impacto do uso da proposta de minimização de congestionamento foi avaliado comparando os cenários com alteração de rotas e sem alteração de rotas. A Figura 7 apresenta o tempo médio de viagem por veículo em cada um dos cenários. No cenário com 300 veículos o tempo médio de viagem por veículo foi cerca de 13% menor quando as estimativas de congestionamento são utilizadas para alteração de rotas. Observa-se

que quanto maior a densidade de veículos, maior o ganho em tempo de viagem com as alterações de rotas.

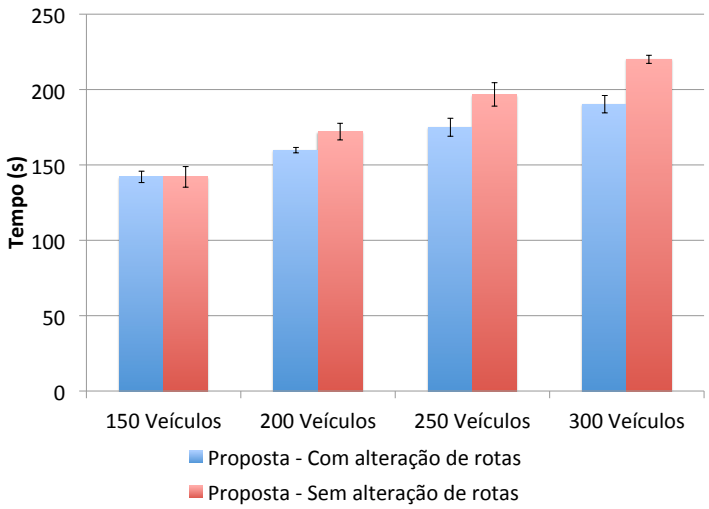


Figura 7. Tempo médio levado por cada veículo para concluir o trajeto.

A Figura 8 apresenta as distâncias médias percorridas pelos veículos nos cenários simulados.

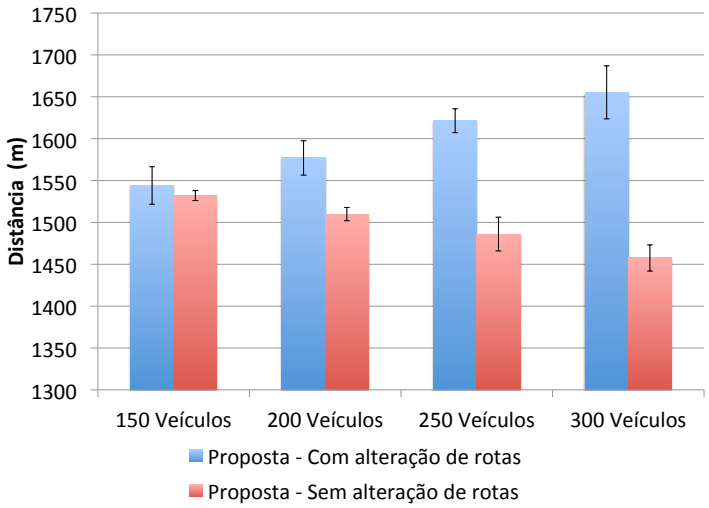


Figura 8. Distância média percorrida por cada veículo durante as simulações.

Observa-se que com as alterações de rotas, os veículos acabam percorrendo uma distância maior. Isso acontece devido aos seguintes fatores: i) as novas rotas sempre possuem uma distância maior ou igual à rota original (já que a rota original foi calculada através do algoritmo de menor caminho de Dijkstra sem considerar pesos de congestionamentos), ii) sem a alteração de rotas, os veículos passam a maior parte do tempo presos nos congestionamentos.

A Figura 9 mostra a porcentagem de veículos que completaram seus percursos em cada um dos cenários.

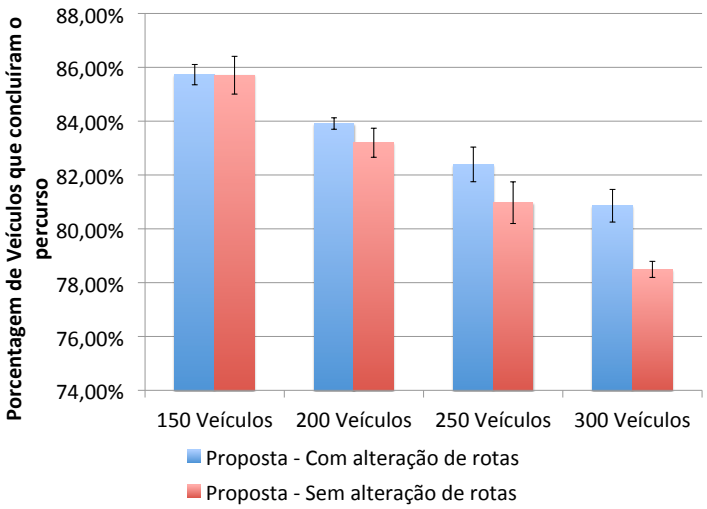


Figura 9. Porcentagem de veículos que completaram o seu percurso.

Observa-se que nos cenários sem alteração de rotas a quantidade de veículos que completaram seus percusos é menor do que nos cenários com alteração de rotas. Sem a alteração de rotas os veículos ficam mais tempo presos em congestionamentos e acabam não concluíndo seu percurso a tempo.

5. Conclusões

Neste trabalho foi proposto um protocolo para estimativa, sinalização e minimização de congestionamentos através de redes veiculares V2V e independente de qualquer infraestrutura adicional. O protocolo permite que cada veículo estime o nível de congestionamento local utilizando a sua velocidade e, através da troca de mensagens com outros veículos, obtenha informações de estimativas de congestionamento de outras vias.

O *overhead* de sinalização do protocolo proposto neste trabalho foi comparado com desempenho do algoritmo proposto em [Milojevic and Rakocevic 2014]. O protocolo proposto apresentou um uso mais eficiente do meio sem fio em todos os cenários avaliados. Em alguns cenários a quantidade de mensagens geradas pelo protocolo proposto chega a ser cerca de 93% menor do que a quantidade gerada pelo protocolo de [Milojevic and Rakocevic 2014]. Também foi avaliada a capacidade do protocolo em minimizar os congestionamentos utilizando as estimativas de congestionamentos para o cálculo de rotas menos congestionadas. Através de simulações observou-se que o protocolo conseguiu atingir o seu objetivo de diminuir o tempo médio de viagem dos veículos.

Como trabalho futuro, pretende-se analisar o desempenho do protocolo em outros cenários. Também pretende-se avaliar o uso do paradigma *store-carry-and-forward* para auxiliar na disseminação de informações em cenários com uma menor densidade de veículos.

Referências

Aissaoui, R., Menouar, H., Dhraief, A., Filali, F., Belghith, A., and Abu-Dayya, A. (2014). Advanced real-time traffic monitoring system based on v2x communications. In *Communications (ICC), 2014 IEEE International Conference on*, pages 2713–2718.

Araújo, G. B., Tostes, A. I. J., de L. P. Duarte-Figueiredo, F., and Loureiro, A. A. F. (2014). Um protocolo de identificação e minimização de congestionamentos de tráfego para redes veiculares. In *XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 207 – 220, Florianópolis.

Cintra, M. (2013). A crise do trânsito em são paulo e seus custos. *GV-executivo*.

dos S. Alves, R., do V. Campbell, I., and de S. Couto, R. (2009). *Redes Veiculares: Princípios, Aplicações e Desafios*, chapter 5, pages 199–254. Sociedade Brasileira de Computação.

Gramaglia, M., Calderon, M., and Bernardos, C. (2014). Abeona monitored traffic: Vanet-assisted cooperative traffic congestion forecasting. *Vehicular Technology Magazine, IEEE*, 9(2):50–57.

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138.

Lomax, T., Turner, S., Shunk, G., Levinson, H. S., Pratt, R. H., Bay, P. N., and Douglas, G. B. (1997). *Quantifying Congestion*, volume 1.

Milojevic, M. and Rakocevic, V. (2014). Distributed road traffic congestion quantification using cooperative vanets. In *Ad Hoc Networking Workshop (MED-HOC-NET), 2014 13th Annual Mediterranean*, pages 203–210.

Rao, A. M. and Rao, K. R. (2012). Measuring urban traffic congestion - a review. *IJTTE*.

Silva, F., Silva, T., Ruiz, L., and Loureiro, A. (2013). Conprova: A smart context provisioning middleware for vanet applications. In *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, pages 1–5.

Sommer, C., German, R., and Dressler, F. (2011). Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15.

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).

Wischoff, L., Ebner, A., Rohling, H., and Halfmann, R. (2003). Sotis - a self-organizing traffic information system. In *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, volume 1, pages 2442 – 2446.

Younes, M. and Boukerche, A. (2013). Efficient traffic congestion detection protocol for next generation vanets. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3764–3768.

Detecção de streamers em redes BitTorrent

Daniel V. C. da Silva, Antonio A. de A. Rocha

Instituto de Computação
Universidade Federal Fluminense (UFF)
Niterói, RJ – Brasil

{dvasconcelos,arocha}@ic.uff.br

Abstract. *Many BitTorrent (BT) clients start use these networks as a video on demand service, using of the popularity and the enormity of the collection of media available. However, transforming the swarms into an on-demand media services can cause serious damage to the overall network performance, all types of users may experience degradation in quality of service, because those streamers clients modify the way to download determined in the protocol. In this paper, a spy Bittorrent client, developed to monitor exchanges of messages in the network is presented. Using the concepts of entropy, is defined a calculation to determine and classify the peers within a swarm. Experiments were made to detect the presence of streamers in public swarms and was also evaluated the impact of this type of user in swarms.*

Resumo. *Muitos clientes BitTorrent (BT) passaram permitir o uso da rede como serviço de mídia sob demanda, fazendo uso da sua popularidade e da enormidade do seu acervo. No entanto, esse uso das redes BT pode trazer sérios prejuízos ao desempenho geral da rede, todos os tipos de usuários podem experimentar queda na qualidade de serviço, isso porque para adaptar essas redes, os clientes modificam a forma de* download *determinada no protocolo. Nesse artigo é apresentado um cliente BitTorrent espião, desenvolvido para monitorar trocas de mensagens na rede. Usando os conceitos de entropia, é definida uma forma de cálculo para determinar e classificar os* peers *dentro de um enxame. Foram feitos experimentos para detectar a presença de* streamers *e também foi avaliado o impacto desse tipo de usuário nos enxames.*

1. Introdução

A melhora na velocidade de acesso à Internet, disponível para os usuários finais nos dias atuais, têm permitido a materialização de novos tipos de aplicação, bem como novas formas de interação com aplicações antigas. Uma prova disso é que, em um passado recente, o tráfego gerado por sistemas P2P (*peer-to-peer*) era responsável por uma fração significativa (mais da metade) de todo o trafego da Internet, sendo que desse montante estimava-se que em torno de 30-50% era proveniente de compartilhamentos de arquivos através do BitTorrent [Menasche et al. 2009]. De acordo com [BitTorrent Inc. 2014], em 2012 a soma mensal de usuários ativos dos *softwares* clientes uTorrent e BitTorrent ultrapassava 150 milhões. Um estudo sobre as diferentes categorias de conteúdo disponibilizados pelos usuários do BitTorrent demonstra que a grande maioria dos conteúdos (aproximadamente 80%) são de arquivos multimídia de filmes, músicas, séries de tv e outros tipos de vídeo ou áudio [Zhang et al. 2010].

Mais recentemente, o aumento da largura de banda da rede de acesso dos usuários na Internet contribuiu também para intensificar o uso das aplicações de mídias contínuas (de áudio e vídeo), também chamadas de aplicações *streaming* multimídia. As taxas mais altas de *download* passaram a permitir que os usuários iniciem a reprodução da mídia com a transferência do arquivo ainda em andamento, isso sem que hajam interrupções ou que pelo menos essas interrupções sejam toleráveis. Assim, dentre os usuários interessados em conteúdos multimídia, essas aplicações passaram a se tornar cada vez mais populares, em detrimento das soluções de compartilhamento de arquivo P2P, como o BitTorrent. Usuários, cada vez mais exigentes, agora desejam (e já podem) iniciar a reprodução da mídia (quase que) instantaneamente e não mais esperar pela conclusão do *download*.

Atualmente, o volume de tráfego gerado pelas aplicações de *streaming* (que incluem NetFlix, Hulu, Youtube, Deezer, Spotify, dentre outras) já é superior ao das aplicações P2P ou de qualquer outro tipo de sistema em uso na Internet. Estudos apresentados em [Cisco 2014] apontam que aproximadamente 66% de todo o tráfego da Internet hoje é oriundo das aplicações de *streaming* de vídeo. Nesse mesmo estudo, os autores prevêem que em 2018 essa fração possa alcançar valores entre 80-90%.

Apesar da crescente popularidade das aplicações de *streaming*, os seus usuários sofrem de dois problemas fundamentias: (i) a falta de escalabilidade inerente à arquitetura cliente/servidor; e, (ii) o acervo limitado de algumas dessas aplicações.

Para lidar com a falta de escalabilidade e, ao mesmo tempo, atender a demanda dos usuários de não ter que esperar a conclusão do *download* de todo o conteúdo para iniciar a reprodução da mídia, sistemas P2PTV podem ser utilizadas. Assim como os sistemas de compratilhamento de arquivo, as aplicações P2PTV surgiram como uma solução escalável para transmissão de vídeo, onde participantes não só assistem a mídia enquanto recuperam o conteúdo, mas também ajudam na transmissão do dados para os demais participantes da rede. Para isso, canais são criados para transmitir o conteúdos de mídia previamente armazenados ou capturados em tempo real (a partir de um sinal de televisão, por exemplo) sem sobrecarregar a máquina do publicador.

Nos últimos anos, uma gama de aplicações P2PTV foram criadas (PPLive, SopCast, CoolStreaming e StreamerP2P são apenas algumas delas). Embora todas elas apresentem soluções para o problema de escalabilidade inerente às arquiteturas com um único servidor central, de alguma forma elas não se tornaram tão populares. Com isso, assim como o NetFlix, Youtube e Hulu, as aplicações P2PTV sofrem com o problema do acervo limitado. Estudos mostram que todas as redes P2PTV somam juntas "apenas" poucas dezenas de milhões de usuários [Hei et al. 2007], sendo a maioria deles originários da China. Por outro lado, apesar da redução significativa do volume de tráfego gerado pelas aplicações BitTorrent nos últimos anos, é indiscutível que ela ainda oferece aos seus usuários um rico acervo de conteúdos de áudio e vídeo.

Nesse cenário, uma abordagem que tem se mostrado promissora é fazer com que os programas clientes BitTorrent explorem a grande popularidade e alta disponibilidade, adaptando essas aplicações para conseguirem operar em modo *streaming*. Para isso, os aplicativos clientes devem alterar as suas política de requisição dos blocos, passando a solicitar as partes do conteúdo de forma sequencial, ao invés de obedecer a política definida pelo protocolo (no caso, a prioridade para o bloco mais raro).

A discussão sobre a possibilidade de co-existência de *peers*, com diferentes políticas de seleção dos blocos em uma mesma rede, não é uma grande novidade [Shah and Paris 2007, Vlavianos et al. 2006, Mendonça and Leao 2012]. Algumas ferramentas já há algum tempo fazem uso desse conceito (conhecidos como *view as you download*). Porém, apenas mais recentemente essa função tem se tornado comum entre as aplicações clientes e mais popular entre os usuários.

Os estudos que tratam da possibilidade de explorar essa capacidade das redes BitTorrent de atender a usuários *streamers*, de modo geral, consideram a ótica dos *peers* interessados na reprodução antecipada e do carregamento sequencial, avaliando apenas fatores como atraso, tempo de reprodução, número e/ou tamanho das interrupções. No entanto, a mudança no mecanismo de seleção dos blocos pode interferir no desempenho da rede como um todo [Li and Zhang 2013]. Estudos apresentados em [Vlavianos et al. 2006, Parvez et al. 2008, Cohen 2003] demonstram que a política de seleção por trechos mais raro levam ao aumento da disponibilidade do conteúdo na rede.

Pensando no funcionamento e desempenho da rede BitTorrent, de uma forma mais ampla, existem portanto algumas questões ainda em aberto, são elas: (i) ainda não há um mecanismo que permita identificar de forma rápida e eficiente usuários operando na função *view as you download* dentro das redes BitTorrent; (ii) não há evidências reais que demonstrem o impacto da presença desse tipo de usuário no desempenho global do sistema; e, (iii) não se sabe ao certo se há hoje um número significativo de usuários conectados às redes convencionais do BitTorrent operando em modo *streaming*. Para responder tais questões, esse trabalho tem como contribuições específicas:

- O desenvolvimento de uma arquitetura de monitoramento da rede BitTorrent, aliada a um método para classificar os usuários conectados a ela como regulares (requisição de blocos mais raros) ou *streamers* (requisição sequencial);
- A análise dos dados coletados em um longo período de monitoramento pela a arquitatura desenvolvida, com o objetivo identificar se há, de fato, uma fração significativa de usuários *streamers* nas redes públicas BitTorrent;
- A realização de experimentos reais, em redes privadas BitTorrent, para mensurar o potencial impacto no desempenho geral do sistema com a presença de clientes operando no modo "*view as you download*".

A organização deste trabalho é feita da seguinte forma. Na Seção 2 é apresentada uma breve revisão teórica, que abordará os conceitos de P2P, *streaming* e entropia. A Seção 3 descreve a arquitetura de monitoramento desenvolvida e os resultados obtidos com a metodologia adotada para detectar a presença de *streamers* em redes BitTorrent. O impacto no desempenho do sistema com a presença de usuários "*view as you download*" é analisada na Seção 4. Por fim, a Seção 5 apresenta as conclusões deste trabalho.

2. Revisão Teórica

Nessa seção serão apresentados os conceitos fundamentais usados na proposta da arquitetura de monitoramento e análises realizadas nesse artigo.

2.1. BitTorrent: do compartilhamento de arquivos ao streaming

As redes P2P se tornaram a principal alternativa à arquitetura cliente-servidor na distribuição de dados. Isso por terem como característica fundamental o fato de que, cada

usuário (nó, *peer* ou par) se comporta simultâneamente tanto como cliente (recebendo os dados) quanto como servidor (servindo os dados já recebidos). Embora o conceito desse tipo de sistema seja bem antigo e usado para outras finalidades que vão muito além do simples compartilhamento de arquivo, nenhuma outra aplicação se tornou tão popular quanto a que foi criada por Bram Cohen [Cohen 2014].

As redes de compartilhamento BitTorrent são usualmente chamadas de "torrent". Nessas redes, os *peers* são denominados *leechers*, se ainda estão baixando conteúdo, ou *seeders*, se já concluíram o *download* por completo. Juntos, os *peers* formam um enxame (do termo em inglês, *swarms*) e contam com uma espécie de coordenador central (conhecido como *Tracker*). A existência do coordenador possibilita que os *peers* conheçam uns aos outros e se conectem para, então, realizar a troca de dados. Para isso, os *peers* devem trocar entre eles os seus respecitvos mapas de *bits* (*bitmaps*) informando aos seus vizinhos os trechos do conteúdo já recuperados.

A disseminação da informação é feita obedecendo uma série de mecanismos definidos no protocolo BitTorrent. Algumas delas são: (i) a política de prioridade, geralmente é implementada de forma que se escolhe o bloco mais raro entre os vizinhos; (ii) a política de compartilhamento "toma-lá-dá-cá", que tenta fomentar a reciprocidade direta entre os *peers* no sistema; e, (iii) a política de compartilhamento otimista, que possibilita a novos usuários receberem inicialmente alguma parte do conteúdo, mesmo sem terem compartilhado nada, para adquirirem algum poder de barganha com os seus vizinhos.

Em [Vlavianos et al. 2006], mostrou-se que dois desses mecanismos do BitTorrent se tornam inadequados quando a aplicação possui a necessidade de uma ordem cronológica no recebimento das partes do conteúdo e/ou uma restrição de tempo na recepção desses dados, como é o caso das aplicações de streaming. A política de prioridade aos pedaços mais raros impõe um modo, de certa forma, aleatório ao recebimento das partes da mídia. Essa mecânica é um problema para aplicações de streaming, uma vez que as partes do conteúdo serão executadas (e assim deveriam também ser recebidas) em sequência. Já a política de compartilhamento "toma-lá-dá-cá" pode impedir que, *peers* com pouca capacidade para contribuir com o sistema (ou seja, taxa de *upload* limitada), recebam os dados a uma taxa satisfatória para a reprodução contínua da mídia.

A Figura 1 ajuda a compreender a necessidade de alteração do protocolo no cliente BitTorrent, para que os usuários possam usufruir da popularidade dos enxames e assistirem enquanto ainda carregam o conteúdo. Essa figura representa a abstração de um enxame BitTorrent com um tracker e três *peers*. Na ilustração, é apresentada uma representação do *bitmap* associado a cada *peer*, onde os blocos de cor cinza indicam os que já foram recuperados e os de cor branca os blocos que ainda estão por receber. Nota-se ainda que, o *peer* com todos os blocos em cinza é um *seeder*, enquanto o *peer* cujo os blocos cinzas são os iniciais é um *leecher streamer* e o outro um *leecher* regular. De acordo com a abstração da figura, o *peer* regular deve escolher aleatoriamente entre o sexto e o sétimo blocos, uma vez que esses blocos são igualmente raros no enxame. No entanto, o *peer streamer* deve necessariamente requisitar o quinto bloco.

Para viabilizar a reprodução de um streaming de vídeo transmitido através do sistema BitTorrent, [Shah and Paris 2007] propõe que sejam alteradas as políticas de seleção dos pedaços e de reciprocidade de transmissão. No entanto, o que tem se visto na rea-

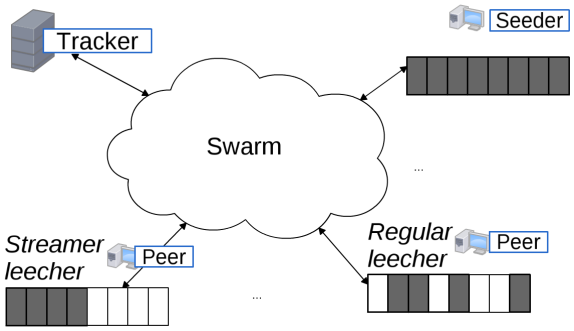


Figura 1: Comportamento dos *peers* aleatórios e sequenciais em um enxame

lidade é que cada aplicação implementa as alterações que seus desenvolvedores acham mais adequadas para a realização do streaming. Nos dias atuais, uma série de clientes BitTorrent têm disponibilizado, integrado à ferramenta, algum tocador de mídia. Isso permite aos usuários assistirem (por completo ou parcialmente) o conteúdo, enquanto este está sendo recuperado. Alguns dos clientes que oferecem essa função são: Vuze, Transmission, Thunder (Xunlei), qBitTorrent, KTorrent, Deluge, Bitcomet e Utorrent. Porém, algumas aplicações vão muito além da simples disponibilização da função "*view as you download*". PopCorn Time e Joker, por exemplo, são dois sistemas que, não só permitem a realização do streaming e execução do conteúdo em um tocador de vídeo, mas tornam todo processo absolutamente simples e totalmente transparente para os seus usuários.

2.2. Entropia Clássica e de Permutação

O conceito de entropia foi adaptado e generalizado pelo matemático americano Claude Shannon em [Shannon 2001] e tem encontrado aplicações nas mais diversas áreas do conhecimento. No contexto de teoria da comunicação (informação), essa teoria é muitas vezes chamada de entropia de Shannon. Em seu trabalho, Shannon define entropia como uma medida de incerteza associada a uma variável aleatória.

Para compreender o conceito, considere um conjunto finito de elementos de uma variável aleatória (digamos um conjunto M, de ordem $n = |M|$), cujo as probabilidades de ocorrência de cada elemento são conhecidas e definidas pela função distribuição de probabilidade discreta $P = \{p_i : i = 1, ..., n\}$. A medida proposta por Shanon, denominada entropia, pode ser expressada da forma:

$$H(M) = -\sum_{1}^{n} p_i \, log \, p_i, \tag{1}$$

onde, sendo considerada a base 2 no logaritmo, a unidade da medida é em bits. Além disso, o limite inferior dessa medida é dado pela máxima certeza sobre a ocorrência do evento, enquanto que o limite superior ocorre em caso de máxima incerteza, ou seja, quando os eventos são equiprováveis. Assim, $H(M) = [0, log_2 \, n]$.

Embora encontre inumeras aplicações, a entropia de Shannon falha por não capturar a relação cronológia de ocorrência dos eventos. Por exemplo, sejam $M_1 = (0, 0, 1, 1)$ e $M_2 = (0, 1, 0, 1)$ dois conjuntos finitos de elementos. Pela definição de Shannon, temos

que $H(M_1) = H(M_2)$. Portanto, a estimativa falha na captura da relação cronologica existente entre os elementos de M. O cálculo da entropia de Shannon também se torna mais complexo à medida que o número de elementos de M aumenta muito.

As limitações impostas ao modelo clássico de entropia são, de certa forma, tratados com uma medida apresentada em [Bandt and Pompe 2002] e revisada por [Riedl et al. 2013]. Denominada Entropia de Permutação, esta medida leva em consideração a ocorrência cronológica dos elementos de um conjunto, comparando os valores adjacentes em uma série temporal.

O método pode ser definido da seguinte forma. Seja s uma determinada ocorrência de uma série de elementos $M = (m_i : i = 1, ..., n)$. Seja v um vetor formado pelos k-ésimos subsequentes a s dessa série. Assim, $v \rightarrow (m_s, m_{s+1}, ..., m_{s+r(k-2)}, m_{s+r(k-1)})$, onde k é chamada a ordem da permutação e determina o tamanho do vetor a ser considerado. O vetor v é associado a um padrão ordinal, denominado permutação $\pi = (r0r1...r(k-1))$ de $(01...k)$, que satisfaz $m_{s+r0} \leq m_{s+r1} \leq ... \leq m_{s+r(k-2)} \leq m_{s+r(k-1)}$. Portanto, cada valor de v é ordenado de forma crescente e o padrão ordinal π é determinado com os deslocamentos dos valores permutados.

Para uma melhor compreensão, considere a sequência $M = (9, 32, 50, 13, 99, 5)$. Para k=3, o vetor correspondente à ocorrência de $s = 1$ é $v = (9, 32, 50)$, sendo ele próprio a sua versão ordenada e, consequentemente, a permutação $\pi = (012)$. Já para a ocorrência de $s = 2$, sendo o vetor $v = (32, 50, 13)$, a ordenação crescente (13,32,50), e resultando em uma permutação $\pi = (201)$.

Note que são $k!$ possíveis combinações de permutações. Seja, então, π_j as frequencias relativas associadas aos j possíveis padrões de permutações, logo $j = 1, ..., k!$. A entropia de permutação de uma sequência M é dada por:

$$H_P(M) = -\sum_{1}^{k!} \pi_j \; log_2 \; p_j, \tag{2}$$

onde $H_P(M) \in [0, log_2 \; k!]$. Logo, a entropia de permutação normalizada (h_P) pode ser obtida por:

$$h_P(M) = -\frac{1}{log_2 \; k!} \sum_{1}^{k!} \pi_j \; log_2 \; p_j. \tag{3}$$

3. Eye of the Swarm

O desenvolvimento de meios que permitam a detecção de aplicações "burlando" o modo padrão de operação do sistema é de suma importância, não só para ajudar a compreender a realidade atual da rede BitTorrent, mas também para auxiliar na criação de mecanismo que permitam inibir tais usuários, se assim for desejado. A questão que se coloca é, será que é possível desenvolver uma técnica não intrusiva (i.e., sem interferir no funcionamento regular do enxame) que permita identificar a existência deste tipo de usuários em um enxame BitTorrent?

Nesta seção, será apresentada uma arquitetura de monitoramento desenvolvida e a metodologia proposta que responde positivamente a questão colocada acima. Dessa forma, é possível analisar o funcionamento e detectar a existência de aplicações BitTorrent operando no modo *streaming* em exames reais.

3.1. Ferramenta Espiã e Metodologia Desenvolvida

Para possibilitar o desenvolvimento da arquitetura de monitoramento e a implementação da metodologia para identificação de *streamers*, uma versão da ferramenta Transmission [TransmissionBT 2014] foi adaptada. Da aplicação foram removidas todas as funções de troca de dados de conteúdo, mantendo apenas as o envio e recebimento dos dados de controle previstos originalmente no protocolo.

O "espião" BitTorrent adaptado funciona como um cliente regular, é visto pelo enxame como tal, porém ele não faz qualquer requisição de conteúdo e registra toda a evolução do *bitmap* (ou *bitfield*) dos *peers* conectados ao enxame. Enquanto se conectar ao maior número possível de *peers* da rede, a ferramenta requisita o mapa de bits de todos os vizinhos e, em seguida, acompanha a evolução dos *bitmaps* através do registro das mensagens de *have* (que são mensagens enviadas por todos os *peers*, para todos os seus vizinhos, informando a conclusão da recuperação de mais um pedaço do conteúdo). Essas informações serão a base da metodologia apresentada a seguir para a classificação dos *peers*.

A partir da evolução do *bitfield* registrado para cada cliente, a técnica proposta usa o conceito de entropia de permutação para determinar se tal aplicação recuperou os trechos de forma sequencial (*streaming*) ou aleatória (a política do bloco mais raro tende a gerar uma recuperação mais uniformemente distribuída). Para isso, a cada cliente registrado com um número suficientemente grande de mensagens de *have* (o experimento descartou dados dos peers com menos de 20 mensagens), uma série M é formada a partir da ordem que foi recebida a informação sobre a recuperação dos pedaços. A partir da sequencia M e para um determinado valor de k, $h_P(M)$ é computado. Valores altos $h_P(M)$ indicam a aleatoriedade da recuperação dos pacotes, enquanto que valores baixos sugerem um modo sequencial de recuperação.

3.2. Validação e Refinamento do Método Proposto

Para validar o método proposto, foram testados os clientes BitTorrent mais populares na Internet. Para cada um desses clientes (um total de nove), foram feitas baterias considerando os modos de *download* aleatório e sequencial (se existisse). Os cenários experimentados foram tanto em ambientes controlados como em ambientes públicos. No primeiro, todas as máquinas encontravam-se em uma rede local e eram controladas . Já no teste em ambiente real, a máquina espiã se conectava a um enxame público, assim como a máquina a ser identificada. O registro da evolução do mapa de bits era feito para a máquina alvo, mas também para todas das demais máquinas encontradas no exame público utilizado.

De posse dos registros de troca de mensagem realizados nas baterias do experimento, considerando ambiente controlado e ambiente real público, foram calculados os valores de entropia por permutação dos programas clientes BitTorrent funcionando em modo aleatório e em modo sequencial. Inicialmente, para esse cálculo foram considerados os valores de k = 2, 3 e 4. No entanto, os resultados demonstram que o valor de k = 2 já é suficiente para a estimativa, além de exigir menos complexidade computacional para ser estimado.

Um dos resultados obtidos de experimentos realizados com o cliente Bitcomet estão ilustrados nas Figuras 2 (a) e (b). Os gráficos indicam a evolução dos valores computados para $h_P(M)$ para $k = 2,\ 3\ e\ 4$ para cada nova mensagem de *have* recebida pelo

espião. Cabe ressaltar que no gráfico é ilustrado uma curva com diversas estimativas de $h(M)$, mas a metodologia prevê a estimativa de apenas um valor para todas as mensagens de "*have*". Neste caso, o objetivo era mostrar a evolução da medida. Pelo gráfico é possível notar que, a partir de um certo tamanho da sequência de *have*, existe uma clara diferença entre os valores computados para $h_P(M)$ sequencial e aleatório, independente do valor de k. (Note que por uma questão de simplicidade, as curvas são identificadas como aleatória ou *streaming* $H(k)$, para $k = 2,\ 3\ e\ 4$). O trecho inicial apresenta a estratégia desse cliente de priorizar o inicio e o fim do arquivo, independente do modo de download.

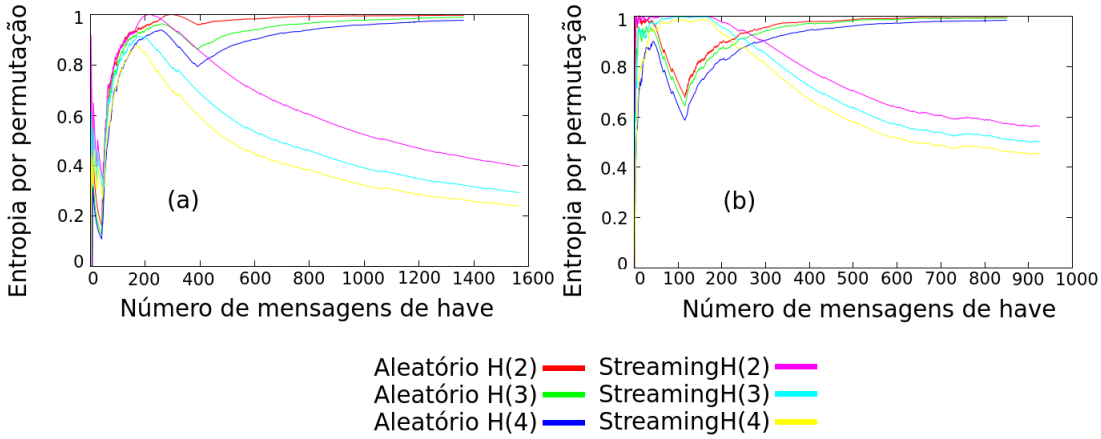


Figura 2: Evolução da entropia por permutação do cliente BitComet em (a) ambiente controlado e (b) em enxame público

A Tabela 1 apresenta todos os valores finais computados. Nela é possível verificar que existe uma grande diferença entre os valores, em ambas estratégias de *download* e entre os clientes. Isso reflete as diferentes implementações adotadas pelos desenvolvedores das aplicações.

Tabela 1: Entropia dos clientes BitTorrent em ambiente controlado

Cliente	Modo	Nº Msg	H(2)	H(3)	H(4)
Bitcomet	Aleatório	163	0.9982	0.9953	0.9856
Bitcomet	Sequencial	1569	0.3966	0.2924	0.2385
BitLord	Aleatório	730	0.9991	0.9969	0.9956
BitLord	Sequencial	604	0	0	0
KTorrent	Aleatório	490	0.9998	0.9985	0.9964
KTorrent	Sequencial	973	0.0465	0.0374	0.0320
qBitTorrent	Aleatório	1550	0.9999	0.9996	0.9988
qBitTorrent	Sequencial	726	0.2422	0.2005	0.1705
Vuze	Aleatório	1570	0.5897	0.5447	0.5039
Vuze	Sequencial	1570	0.5539	0.5045	0.4619
Xunlei	Aleatório	3139	0.4338	0.3986	0.3749
Xunlei	Sequencial	1568	0.1975	0.1645	0.1483

Muito embora seja clara a existência da diferença entre a entropia calculada para

o caso sequencial e o caso aleatório, não há um *threshold* comum entre todas as diferentes versão que possa ser usado para determinar a classificação dos *peers*. Por isso, a solução adotada passa por atribuir a cada tipo de aplicação estudada um limiar diferente. Esse adequação pode ser feita, pois nas mensagens de controle enviadas pelos clientes é indicado nome da aplicação remota. Certamente, a razão pela qual diferentes aplicações possuem variações significativas no valor computado da entropia de permutação está relacionado às opções feitas pelo grupo de desenvolvedores na implementação das ferramentas.

3.3. Arquitetura de Monitoramento e Experimentos em Larga Escala

Com o objetivo de identificar a existencia (e qual a fração) de usuários operando em modo *streaming* em enxames públicos, experimentos em larga escala foram realizados. A arquitetura de monitoramento, resumida pela Figura 3, operou por noventa dias consecutivos, da seguinte forma: (i) a cada duas horas eram obtidos os links dos 100 enxames de vídeo mais populares do PirateBay; (ii) para cada um desses 100 enxames, um processo espião era disparado para realizar o monitoramento (coleta de dados) de todos os *peers* conectados ao enxame; (iii) Ao final do período de duas horas, os logs de dados de controle eram armazenados em uma base de dados, os demais registros e metadados apagados do monitor, e o processo de monitoramento voltava para a etapa (i).

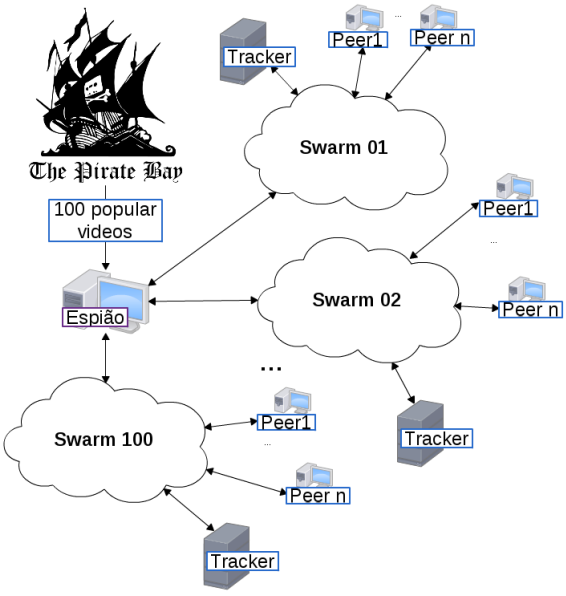


Figura 3: Esquema do experimento em enxames populares do PirateBay

O espião se conectou a mais de 1 milhão e meio de *peers*, de um total de mais de 100 mil enxames. A coleta resultou em uma base com aproximadamente 40 gigabytes de registro de mensagens.

Para todos os *peers* monitorados por uma fração de tempo suficientemente grande, foi calculado a entropia de permutação. A Tabela 2 apresenta um resumo das coletas e classificações realizadas pela arquitetura de monitoramento por ferramenta cliente. Dos números mostrados, nota-se que dos mais de 1.5 milhões de *peers*, apenas aproximadamente 7 mil deles (cerca de 0,53%) foram classificados como *streamers*.

Tabela 2: Avaliação da existência de *streamers* em duas etapas

Cliente	Nº Peers	Streamers	%
Vuze	145285	657	0,45%
Transmission	202439	1	0,00%
Thunder (Xunlei)	2346	376	16,03%
qBitTorrent	7834	9	0,11%
KTorrent	62	0	0,00%
Deluge	28246	0	0,00%
BitTorrent	97761	687	0,70%
BitLord	181	0	0,00%
Bitcomet	7110	29	0,41%
Utorrent	764185	5212	0,68%
IL50	5791	0	0,00%
JS09	45	0	0,00%

3.4. Monitoramento de Enxames Específicos

Não é incoerente imaginar que usuários sequenciais são pessoas que não desejam esperar para desfrutar do conteúdo. Então, uma hipótese é que esse tipo de indivíduo acessa às redes assim que uma determinada mídia de interesse é disponibilizada na Internet, e possivelmente faria uso de um modo de operação "*view as you download*". Para verificar essa hipótese, a ferramenta espiã foi também configurada para monitorar por 24 horas, os enxames dos últimos 5 episódios da quarta temporada da série Walking Dead e todos os episódios da quarta temporada de Game of Thrones, assim que esses vídeos aparecessem na lista de mais populares do PirateBay.

No caso do Game of Thrones, o espião se conectou a 110 mil *peers*, e dentre esses 3600 foram classificados como *streamers*, o que representa 3,28% do total de usuários monitorados. Embora não seja um número muito alto, é um aumento considerável, quando comparado ao valor inicial de 1,10% do experimento realizado com todos os enxames de vídeo populares do PirateBay. Já o experimento com a série Walking Dead, o espião se conectou a 13143 *peers*, e dentre esses apenas 57 possuem entropia por permutação que os classifiquem como *streamers*, o que representa 0,43% dos *peers*. Um valor muito próximo ao do experimento em larga escala.

Um resultado que merece destaque é ilustrado na Figura 4. Nela são mostrados os valores ordenados da entropia de permutação dos *peers*, isso para um tipo específico de cliente BitTorrent (Xunlei Thunder). O comportamento do gráfico ilustra claramente a separação de usuários *streamers* dos usuários em modo aleatório. Nota-se que essa ferramenta em especial, para essa série, é muito comum encontrarmos usuários operando no modo *"view as you donwload"*. Neste caso são mais de 1000 *streamers*, de um total aproximado de 1600 *peers* (63%).

4. Análise do Impacto de *Streamer*s no Desempenho dos Enxames

Bram Cohen defende argumenta em [Cohen 2003] que a robustez das redes BitTorrent é uma consequencia direta da política apropriada de seleção de blocos mais raros. Porém, as discussões são superficiais e, nem tão pouco, o autor apresenta evidências reais que

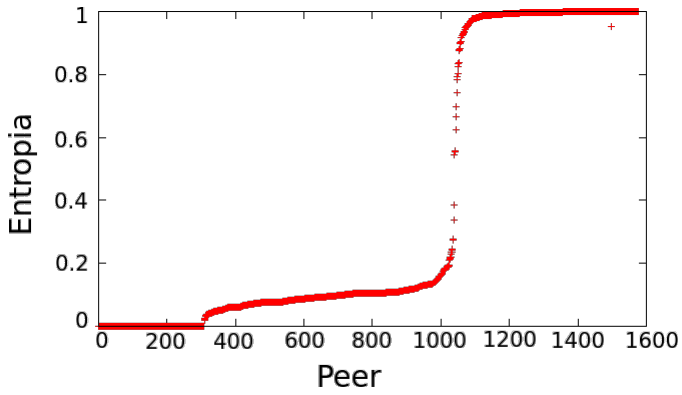


Figura 4: Entropia por permutação de clientes Xunlei (Game of Thrones)

garantam essa afirmação. Nessa seção serão apresentados experimentos feitos em laboratório, considerando diversos cenários. O objetivo é entender se de fato a existência de usuários *streamers* interferem no desempenho global dos enxames. E, se sim, a partir de que fração de usuários na rede isso se torna mais significativo. Para isso, foi considerado o tempo de *download* como principal medida de desempenho do enxame.

Foram definidos três diferentes cenários, todos com 0%, 10%, 20%, 30%, 40% e 50% de usuários em modo sequencial e *peers* com velocidade limitada a 300kB. Todos os *peers* saem do enxame assim que concluírem o seus *download*. Esses três conjuntos de experimentos diferem fundalmentalmente entre eles pelo padrão de entrada considerado para os *peers* e o tamanho da vizinhança, como descritos a seguir:

- Entrada dos *peers leechers* em rajada, sem restrições quanto ao número de conexões simultâneas que os *peers* podem estabelecer.
- Entrada dos *peers leechers* em rajada, sendo que o tamanho da vizinhança de cada *peer* foi limitada a 10 vizinhos.
- Entrada dos *peers leechers* a cada 10 segundos, com o tamanho da vizinhança de cada *peer* limitada a 10 vizinhos.

A Figura 5 ilustra a média dos valores da sessão com vizinhança ilimitada e entrada simultânea. Ela apresenta, de forma direta, o comportamento do tempo de *download* durante as seis baterias. No gráfico é possível identificar uma redução da média do tempo de download para a bateria 10%, porém existe um aumento no tempo total. A bateria de 20% tem um melhor desempenho geral, pois embora sua média e moda sejam um pouco maiores que a da bateria anterior, o tempo total do experimento é menor que a metade de todos os outros. E por fim, nas baterias a com 30% ou mais, existe um clara piora nas métricas de tempo.

A Figura 6 apresenta a média dos valores da sessão com vizinhança limitada e entrada simultânea. Embora as curvas sejam mais suaves, fica claro que o comportamento observado na configuração anterior e apresentado na Figura 5 se repete nessa configuração. É perceptível um aumento do tempo total na bateria a 10%, e embora não exista uma redução da média do tempo a 10%, ela também não aumenta muito, comparativamente. Também percebe-se que a bateria de 20% tem um melhor desempenho geral. Já nas baterias a com 30% ou mais, existe um clara piora nas métricas de tempo, embora não muito direta em 30%.

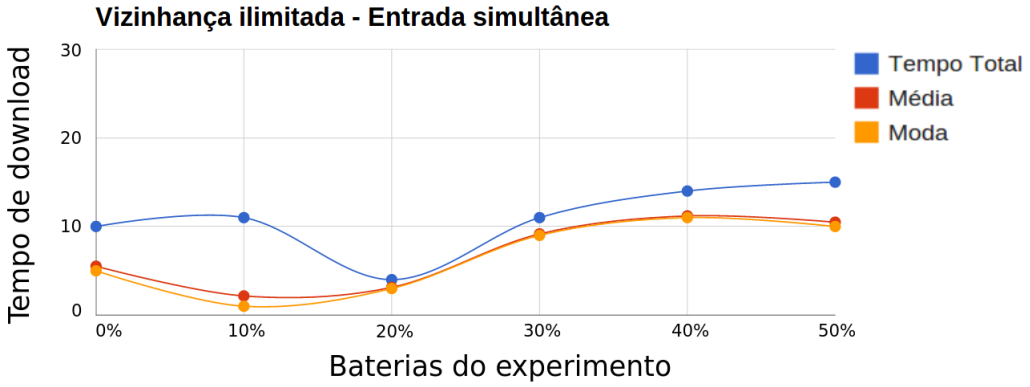


Figura 5: Desempenho do enxame com vizinhança ilimitada e entrada simultânea

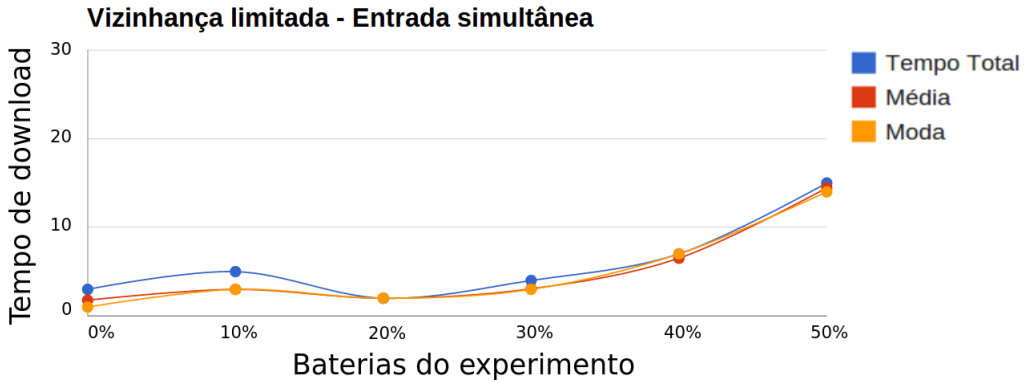


Figura 6: Desempenho do enxame testado em laboratório com vizinhança limitada e entrada simultânea

No exemplo de sessão com vizinhança limitada e entrada dos *peers* a cada 10 segundos (apresentada pela Figura 7) é possível identificar um comportamento um pouco diferente dos resultados anteriores. Nessa configuração, a bateria de 10% tem um resultado pior que as de 0%, 20%. 30% e 40%, e a bateria a 50% continua apresentando o pior desempenho geral.

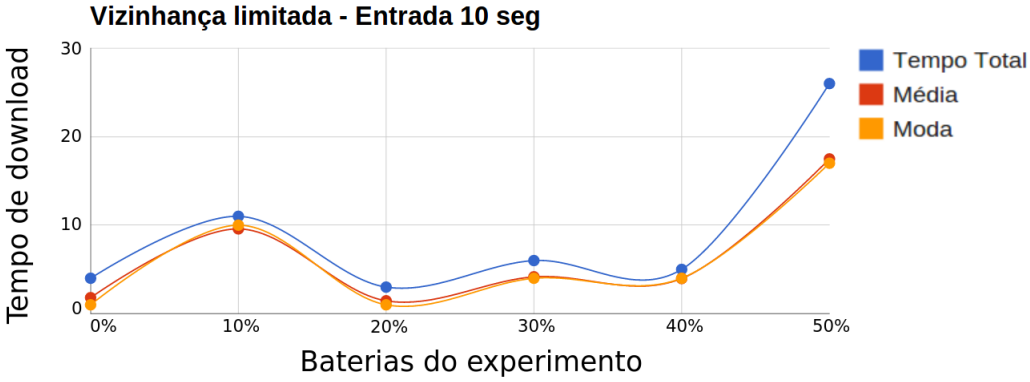


Figura 7: Desempenho do enxame testado em laboratório com vizinhança ilimitada e entrada a cada 10 segundos

Considerando as diferentes configurações de experimento, é possível afirmar que existe sim uma piora no tempo de *download* no enxame, seja ele individual ou total, à medida que se cresce o número de usuários operando em modo sequencial nessas redes. Em casos extremos, a piora no desempenho visto em uma bateria sem a presença de *streamers* se comparado a uma com metade dos *peers* operando em modo sequencial chega a ser 17 vezes maior, e o tempo total da bateria chegou a ser 6 vezes pior com 50% dos *peers* em modo *streaming*. Esse trabalho não apresenta intervalo de confiança pois os dados mostram pequena variância.

Por outro lado, os resultados apresentados também sugerem que, para alguns parâmetros específicos, a presença desses usuários pode ser irrelevante ou mesmo até bem vinda, desde que não ultrapasse um limite definido pela configuração do enxame em questão.

5. Conclusões

A adaptação das aplicações BitTorrent para uso de mídia sob demanda tem se apresentado cada vez mais frequente. Só no último ano duas novas ferramentas foram apresentadas na Internet com grande repercursão. Sem dúvida, esse movimento foi motivado pelo vasto acervo de conteúdo existente no BitTorrent, somado ao alto número de usuários desse modelo de aplicação. Além, é claro, de um aumento da demanda de usuários por soluções de vídeo sobre demanda.

A questão fundamental que este trabalho tenta responder são: Será que existe hoje um número significativo de usuários *streamers*? Será que é possível identificar em tempo real a existência desse tipo de usuário em enxames reais? A presença desses usuários burlando o sistema pode afetar o desempenho global da aplicação?

Esse trabalho, usando como base os conceitos de entropia por permutação, propos uma metodologia que permita clasisficar os usuários como *streamers* ou aleatórias. Além disso, uma arquitetura de monitoramento desenvolvida permitiu realizar experimentos em larga escala e identificar que, embora não muito alta, já há uma fração relevante de usuários *streamers* usando as redes BitTorrent para baixar conteúdos de forma sequencial. Por fim, experimentos realizados em laboratório permitiram verificar que existe valores aceitáveis da fração de usuários sequenciais dentro dos enxames, sem que haja degradação da qualidade experimentada pelos usuários. Porém, para um número elevado (acima de 30%), em geral há uma perda significativa do desempenho para todos no sistema.

Esse trabalho aponta, como possibilidades de estudos futuros, a ampliação da análise do impacto de usuários sequenciais nos enxames. Mesmo com os experimentos realizados para avaliar o impacto não se foi capaz de determinar alguns comportamentos da rede, como por exemplo a degradação após os 20%.

Referências

Bandt, C. and Pompe, B. (2002). Permutation entropy: a natural complexity measure for time series. *Physical Review Letters*, 88(17):174102.

BitTorrent Inc. ((acessado em dezembro de 2014)). Bittorrent and utorrent software surpass 150 million user milestone; announce new consumer electronics partnerships. http://www.bittorrent.com/intl/es/company/about/ces_2012_150m_users.

Cisco ((acessado em dezembro de 2014)). Cisco visual networking index: Forecast and methodology, 20132018. "http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf".

Cohen, B. (2003). Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72.

Cohen, B. ((acessado em agosto de 2014)). The bittorrent protocol specification. http://www.bittorrent.org/beps/bep_0003.html.

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.

Li, Z. and Zhang, T. (2013). Understanding the quality-of-service of ubiquitous bittorrent-like media streaming. *Greenorbs*.

Menasche, D., Rocha, A., Li, B., Towsley, D., and Venkataramani, A. (2009). Content availability and bundling in swarming systems. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 121–132. ACM.

Mendonça, G. G. and Leao, R. M. (2012). Btstream–um ambiente para desenvolvimento e teste de aplicaç oes de streaming p2p. *XXX SBRC, Salao de Ferramentas, Ouro Preto, MG, Brazil*.

Parvez, N., Williamson, C., Mahanti, A., and Carlsson, N. (2008). Analysis of bittorrent-like protocols for on-demand stored media streaming. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):301–312.

Riedl, M., Müller, A., and Wessel, N. (2013). Practical considerations of permutation entropy. *The European Physical Journal Special Topics*, 222(2):249–262.

Shah, P. and Paris, J.-F. (2007). Peer-to-peer multimedia streaming using bittorrent. In *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, pages 340–347. IEEE.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55.

TransmissionBT ((acessado em dezembro de 2014)). Transmission. http://www.transmissionbt.com/.

Vlavianos, A., Iliofotou, M., and Faloutsos, M. (2006). Bitos: Enhancing bittorrent for supporting streaming applications. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6. IEEE.

Zhang, C., Dhungel, P., Wu, D., and Ross, K. W. (2010). Unraveling the BitTorrent ecosystem. *IEEE Transactions on Parallel Distributed Systems*, 22(7):1164–1177.

Analisando o Impacto de Compartilhamentos no Dropbox em uma Rede Acadêmica *

Glauber Dias Gonçalves[1], Idilio Drago[3], 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

[3]Politecnico di Torino - Italy

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

idilio.drago@polito.it; alex.borges@ufjf.edu.br

Resumo. *Serviços de armazenamento na nuvem (e.g., Dropbox) são meios populares de compartilhamento de conteúdo e realização de trabalho colaborativo. Contudo, o compartilhamento nas nuvens pode levar a desperdício de largura de banda quando o mesmo conteúdo é recuperado de servidores remotos por múltiplos usuários em um mesmo domínio de rede. Este artigo apresenta uma caracterização dos padrões de compartilhamento no Dropbox a partir de dados de tráfego coletados de um campus universitário durante 4 meses. Identificamos que o volume compartilhado via Dropbox é razoavelmente alto. Em seguida, utilizamos os resultados da caracterização para desenvolver um gerador de cargas sintéticas que permite avaliar alterações no protocolo de sincronização do Dropbox. Propomos então uma arquitetura de sincronização que inclui caches para temporariamente armazenar as atualizações dos usuários. Nossos resultados indicam que, mesmo com um cache pequeno, é possível evitar praticamente todos os downloads redundantes, o que beneficia provedores do serviço de armazenamento, usuários e a Internet.*

Abstract. *Cloud storage services (e.g., Dropbox) are a popular means for sharing content and performing collaborative work. Yet, content sharing via the cloud might result in bandwidth wastage when repetitive data is downloaded by different users in the same network domain. This paper first characterizes sharing patterns in Dropbox by analyzing data collected from a campus network for 4 months. We identify that the volume of data sharing in such homogeneous environment is reasonably high. Next, we use the characterization results to implement a synthetic workload generator that allows us to test alternatives for the Dropbox synchronization protocol. We then propose a synchronization architecture that includes network caches to temporarily hold user updates. Our evaluation of the proposed solution indicates that, even with a small cache, it is possible to achieve almost the maximum possible reduction of downloads from remote servers, thus benefiting storage providers, end users and the Internet.*

*Esta pesquisa é financiada pelo INCTWeb (MCT/CNPq 573871/2008-6), CNPq, CAPES e FAPEMIG.

1. Introdução

Os serviços de armazenamento pessoal em nuvem (*Personal Cloud Storage*) estão cada dia mais populares, tanto entre usuários domésticos quanto corporativos. Isso pode ser observado, por exemplo, pela crescente popularidade de termos como "Dropbox", "iCloud" e "Google Drive" em consultas ao principal serviço de buscas da Internet (i.e., Google)[1]. Serviços de armazenamento nas nuvens oferecem diversas vantagens a seus usuários. Esses podem eliminar praticamente toda a sua infraestrutura de armazenamento, confiando somente em servidores remotos. Além disso, a migração para esse tipo de serviço não requer grandes investimentos. Usuários passam a contar com replicação (i.e., *backup*) e compartilhamento de dados via Internet de forma transparente e flexível.

Serviços de *Personal Cloud Storage* também vêm atraindo a atenção de pesquisadores, principalmente pelo impacto gerado no tráfego da Internet e em redes locais [Li et al. 2014, Gracia-Tinedo et al. 2013]. Dentre os serviços existentes, o Dropbox vem sendo um dos mais estudados, em particular por sua grande inserção em ambientes acadêmicos. Em [Drago et al. 2012] foi observado um volume significativo de tráfego relacionado ao Dropbox em campi universitários. Tal volume foi comparado ao tráfego de serviços tradicionais da Internet, como o YouTube.

O compartilhamento de arquivos no Dropbox pode levar a desperdício de banda quando um mesmo conteúdo é recuperado dos servidores remotos pelos vários usuários que o compartilham. O Dropbox busca otimizar a utilização da rede por meio de vários mecanismos em seu cliente, incluindo compressão de dados, de-duplicação e agrupamento de transferências (i.e., *bundling*) [Drago et al. 2013]. Porém, embora esses mecanismos sejam eficientes em reduzir o tráfego particularmente das transferências dos clientes para a nuvem (i.e., *upload*), eles tem um efeito limitado na eliminação das redundâncias geradas pelo compartilhamento de conteúdo, que aparecem no *download* de dados. De fato, evidências [Drago et al. 2012] mostram que o volume de *download* é aproximadamente duas vezes superior ao de *upload* em redes acadêmicas.

Um dos principais mecanismos do Dropbox para minimizar essas redundâncias é o protocolo `LAN sync`[2]. Por esse mecanismo, arquivos compartilhados entre dispositivos em uma mesma (sub-)rede (i.e., em um mesmo perímetro de *broadcast*) são atualizados sem que cada um dos dispositivos recupere os arquivos da nuvem. Todavia, esse mecanismo não explora todo o potencial de redução do uso da rede. Isso porque grande parte das redes, como as redes acadêmicas, são compostas por várias sub-redes. Os dispositivos em sub-redes distintas não se beneficiam do protocolo `LAN sync`. Logo, usuários de um mesmo domínio (e.g., uma universidade) que compartilham conteúdo, mas cujos dispositivos estejam em sub-redes diferentes, devem recuperar o (mesmo) conteúdo diretamente da nuvem, gerando tráfego duplicado e desperdício de banda.

Nesse trabalho, investigamos os padrões de compartilhamento entre usuários do Dropbox localizados em um mesmo domínio buscando responder às seguintes questões: *O volume de tráfego gerado pelo download de conteúdo compartilhado por usuários locais ao domínio é significativo? É possível reduzir o desperdício de banda associado a esse tráfego de forma efetiva e eficaz com o uso de caching?* Apesar dos estudos sobre

[1] http://www.google.com/trends

[2] `https://www.dropbox.com/pt_BR/help/137`

padrões de carga e tráfego do Dropbox [Costa et al. 2014, Gonçalves et al. 2014], essas questões ainda não foram abordadas.

Para tal, primeiramente caracterizamos os padrões de compartilhamento entre usuários do Dropbox em um grande campus universitário, a partir de dados de tráfego coletados durante quatro meses. Nesse ambiente, o Dropbox representa um grande volume de dados [Duarte et al. 2015]. A partir dessa caracterização, desenvolvemos e validamos um gerador de cargas sintéticas capaz de capturar, de forma realista, o comportamento dos compartilhamentos pelos usuários locais no Dropbox. Por fim, propomos uma mudança na arquitetura do protocolo de sincronização do Dropbox que considera a inclusão de *caches* locais ao domínio para armazenar as atualizações feitas pelos usuários.

Nossos resultados indicam que o volume de dados compartilhados entre os usuários do campus via Dropbox é razoavelmente alto: cerca de 24% de todo o tráfego de *download* gerado pelo Dropbox está associado ao compartilhamento de conteúdo entre 2 ou mais usuários. Mais ainda, mostramos (via simulação) que a nossa proposta de arquitetura de *cache* pode reduzir o desperdício de banda. As simulações, conduzidas a partir de cargas sintéticas providas pelo nosso gerador, mostram que, mesmo com um cache pequeno, seria possível evitar praticamente todos os *downloads* redundantes ocasionados por compartilhamentos locais. Por exemplo, um *cache* de apenas 100 GB, em um cenário de carga alta, resultaria em aproximadamente 93% de redução no volume de *downloads* em compartilhamentos (i.e., *byte hit ratio*). Tal redução, que se reflete em economia de banda na(s) rede(s) entre os servidores de armazenamento e o ponto de entrada da rede universitária, beneficiaria tanto o provedor do serviço de armazenamento (Dropbox) e seus usuários, quanto a Internet como um todo.

Em suma, as principais contribuições deste artigo são: (1) a primeira caracterização dos padrões de compartilhamento entre usuários de um mesmo domínio; (2) um gerador de cargas sintéticas que captura, de maneira realista, os padrões de compartilhamento e atualização de conteúdo por usuários do Dropbox em um campus universitário; e (3) a proposta e a avaliação de uma mudança na arquitetura do protocolo do Dropbox, que considera a inclusão de *caches* locais ao domínio, resultando em considerável economia de banda. Ressaltamos que o gerador desenvolvido será disponibilizado como software livre e é, portanto, uma ferramenta útil para avaliações do Dropbox e/ou de modificações da sua arquitetura em outros cenários.

A seguir, a Seção 2 discute trabalhos relacionados. A Seção 3 apresenta conceitos básicos do Dropbox e a metodologia adotada para coleta e análise de dados. A Seção 4 discute resultados da caracterização dos compartilhamentos, bem como o gerador de cargas sintéticas desenvolvido. Nossa proposta de arquitetura de sincronização e *caching* é detalhada na Seção 5. Por fim, a Seção 6 sumariza o artigo.

2. Trabalhos Relacionados

Este trabalho é pioneiro por avaliar o impacto, no tráfego de uma rede particular (especificamente, uma rede universitária), do compartilhamento de conteúdo por meio de clientes de armazenamento nas nuvens. O trabalho de [Costa et al. 2014] avalia alguns aspectos do compartilhamento entre usuários do Dropbox usando uma coleta de dados de voluntários. Nosso trabalho, ao invés disso, estuda os compartilhamentos a partir de informações do protocolo de controle do Dropbox obtidas por medições passivas. Isso nos

permite apresentar uma análise mais abrangente tanto em termos do tamanho da amostra (i.e., cobrindo a população *completa* de uma rede acadêmica), quanto da duração da amostra (i.e., cobrindo vários meses).

A caracterização feita neste trabalho e a construção do nosso gerador de cargas sintéticas tomam como base resultados de [Drago et al. 2012], que apresenta uma caracterização do tráfego do Dropbox a partir de medidas passivas, e de [Gonçalves et al. 2014], que propõem um modelo para o funcionamento do cliente Dropbox. Diferentemente desses trabalhos, nós caracterizamos os padrões de compartilhamento no Dropbox, desenvolvemos um gerador de cargas sintéticas, e propomos uma arquitetura de sincronização para auxiliar o serviço de armazenamento na nuvem.

Alguns trabalhos [Mager et al. 2012, Wang et al. 2012] também analisam o tráfego de ferramentas de armazenamento nas nuvens, como o Dropbox ou o Wuala, a partir de medições de tráfego na rede. As características dos arquivos armazenados em serviços similares ao Dropbox e os possíveis gargalos em seus protocolos são estudados por [Li et al. 2014, Liu et al. 2013]. Nenhum desses trabalhos, porém, discute o impacto de compartilhamentos no tráfego dos serviços.

A Qualidade de Experiência (QoE) de serviços de armazenamento foi estudada por [Amrehn et al. 2013, Casas and Schatz 2014]. Diversos aspectos técnicos como a largura de banda disponível e o tempo de sincronização são identificados como parâmetros importantes para a satisfação dos usuários. Nosso trabalho propõe um gerador de cargas sintéticas realistas que pode ser usado para quantificar o impacto de mudanças em protocolos e clientes de armazenamento na QoE. Além disso, propomos e validamos uma mudança na arquitetura do protocolo de sincronização do Dropbox, que mostramos ter potencial para reduzir o tráfego em redes com alto grau de compartilhamento (e.g., campi universitários) – logo, com provável impacto positivo na QoE percebida pelos usuários.

Por fim, alguns trabalhos comparam o desempenho de ferramentas [Drago et al. 2013, Hu et al. 2010, Gracia-Tinedo et al. 2013], serviços e provedores [Li et al. 2010] de armazenameno na nuvem. Nosso trabalho trata de desempenho sob uma perspectiva diferente. Nós analisamos uma característica importante desses serviços, i.e., o compartilhamento de arquivos, visando uma melhoria de desempenho, a partir da diminuição do desperdício de banda com *downloads* redundantes.

3. Conceitos e Metodologia de Coleta de Dados

3.1. Dropbox: Visão Geral

Cada usuário do Dropbox tem um repositório associado à sua conta. Esse repositório é composto minimamente por uma *pasta raiz*, onde o usuário pode armazenar arquivos ou criar subpastas. A aplicação também oferece ao usuário a funcionalidade de compartilhar arquivos. Para tal, o usuário pode configurar uma ou mais *pasta(s) compartilhada(s)* em seu repositório e convidar outros usuários a terem acesso ao seu conteúdo. Usuários convidados recebem notificações por e-mail. Caso aceitem o convite, eles terão a pasta compartilhada adicionada aos seus dispositivos, podendo então fazer atualizações. Tecnicamente, o Dropbox identifica tanto pastas compartilhadas quanto pastas raiz da mesma maneira – pastas compartilhadas simplesmente aparecerem no dispositivo de vários usuários. Por esse motivo, utilizamos os termos *pasta*, *pasta raiz* e *pasta compartilhada* sem nenhuma distinção neste trabalho.

A arquitetura do Drobpox apresenta dois componentes principais: (1) *servidores de controle*, que são gerenciados pelo Dropbox em sua infraestrutura privada; e (2) *servidores de armazenamento*, que são hospedados na Amazon. Os arquivos de usuários são sempre armazenados na nuvem pública da Amazon. Tanto para servidores de controle quanto de armazenamento, subdomínios de `dropbox.com` são usados para identificar os diferentes componentes do serviço e as funcionalidades específicas do cliente Dropbox [Gonçalves et al. 2014, Drago et al. 2012].

Quando iniciado, o cliente Dropbox se registra em um servidor de controle e requisita a lista de atualizações no repositório do usuário. Em seguida, o cliente faz o *upload/download* dessas atualizações para/de os servidores de armazenamento. Toda comunicação para autenticação e troca de dados é criptografada [Drago et al. 2012]. Porém, o cliente mantém uma conexão TCP com um servidor de controle, na qual mensagens não-criptografadas do sistema de arquivos do Dropbox são trocadas, informando o estado de cada pasta compartilhada no dispositivo do usuário. O cliente Dropbox troca mensagens (via protocolo HTTP) com servidores de controle a uma taxa aproximada de 1 requisição por minuto. Essas mensagens são cruciais para o nosso trabalho como descrito a seguir, por conter informações atualizadas do estado do repositório do usuário e da versão atual de cada pasta compartilhada pelos usuários.

3.2. Coleta de Dados

Nós coletamos traços de dados de tráfego da rede de um campus universitário no Brasil com cerca de 57 mil pessoas, que inclui tráfego gerado por laboratórios e escritórios de administração, conforme a metodologia proposta em [Drago et al. 2012]. A ferramenta `tstat` [Finamore et al. 2011] foi usada para coletar informações sobre as conexões TCP nessa rede e as requisições HTTP direcionadas aos servidores do Dropbox. Foram utilizadas as mesmas heurísticas de [Drago et al. 2012] para identificar e classificar o tráfego Dropbox. Por exemplo, para classificar o tráfego entre diferentes funcionalidades do Dropbox (i.e., controle e armazenamento de dados), procuramos por textos como `*.dropbox.com` nos certificados TLS/SSL ou no domínio que os clientes requisitaram aos servidores DNS. Como resultado desta etapa, obtivemos uma base de dados contendo informações sobre cada conexão TCP direcionada ao Dropbox, incluindo a funcionalidade provida pelo servidor específico contactado pelo cliente.

Diferentemente das coletas feitas em [Drago et al. 2012, Gonçalves et al. 2014], nossa coleta inclui todas as mensagens de controle do sistema de arquivo do Dropbox transferidas em conexões não-criptografadas. Essas mensagens de controle incluem: (1) o endereço IP anonimizado do dispositivo conectado ao Dropbox; (2) um identificador único do dispositivo; (3) um código numérico associado ao usuário; (4) um identificador único de cada pasta compartilhada disponível no dispositivo; e, finalmente, (5) um identificador da versão atual de cada pasta compartilhada no dispositivo[3]. Importante para a nossa análise, o número da versão de cada pasta compartilhada é um inteiro crescente, o que nos permite monitorar a evolução das pastas ao longo tempo.

Dessas informações de controle do Dropbox, podemos identificar eventos de modificações em pastas compartilhadas, como o aparecimento e o desaparecimento de

[3]Nossa coleta não inclui nenhuma informação que possa revelar a identidade dos usuários, bem como informações pessoais ou dos arquivos armazenados no Dropbox.

Tabela 1. Sumário do tráfego Dropbox processado dos traços da rede.

Período	24/03–31/07/14
Volume Upload (TB)	1.97
Volume Download (TB)	4.39
Número de usuários	6478
Número de usuários que compartilham pastas	3445
Número de pastas	16485
Número de pastas compartilhadas localmente	3233
Número de notificações de modificação em pastas	3160095

compartilhamentos, ou a alteração de arquivos. Para cada um desses eventos, os servidores Dropbox anunciam um novo número de versão da pasta para todos os dispositivos participantes do compartilhamento, que contactam servidores de dados do Dropbox para obter as alterações. Como as transferências de dados são criptografadas, registramos apenas (1) o endereço IP anonimizado do dispositivo; (2) os instantes de tempo inicial e final; e (3) o volume de dados transferido. Por fim, analisando o volume de dados, podemos imediatamente identificar se o usuário criou um novo conteúdo (i.e., houve um *upload*) ou se ele recebeu modificações executadas em outro dispositivo (i.e., houve um *download*).

Portanto, extraímos dos traços coletados: (1) as notificações de modificações nas pastas (mensagens de controle); e (2) informações sobre as transferência de dados (mensagens de dados). Dado que essas informações são coletadas de forma independente e não contem um identificador único comum (e.g., o identificador do dispositivo), propomos um método para associar as notificações aos dados transferidos. Por meio de experimentos em ambiente controlado, primeiramente, observamos que o cliente emite notificações imediatamente após modificar uma pasta (*upload*) ou antes de receber uma modificação (*download*). A seguir, medimos o tempo decorrido entre transferências de dados e notificações mais próximas, ambas pertencentes ao mesmo endereço IP nos traços de dados. Observamos que a distribuição desse tempo apresenta um ponto de inflexão por volta de 10 segundos. Logo, adotamos esse limiar para realizar a associação de notificações aos dados transferidos. As transferências que ocorreram em um intervalo de tempo para as notificações mais próximas, superior a esse limiar, foram descartadas, pois podem levar a associações errôneas. Analisamos o impacto dos descartes na próxima seção.

A Tabela 1 sumariza os dados coletados, apresentando o período da coleta, o volume de tráfego e os números totais de usuários, pastas e modificações observados. Mostramos na tabela o número de pastas que são compartilhadas por vários usuários locais, assim como o número de usuários (locais) que compartilham essas pastas. Note que eventuais compartilhamentos com usuários externos à rede monitorada não são contabilizados nas estatísticas de pastas compartilhadas da Tabela 1. Também são descartadas notificações feitas em pastas diferentes por usuários diferentes sob um mesmo NAT, desde que as mesmas ocorram em um intervalo de ate 10 segundos. Tais notificações representam apenas 1% dos dados.

4. Avaliação dos Compartilhamentos no Dropbox

Esta seção discute os padrões de compartilhamento de conteúdo no Dropbox. Avaliamos o volume dos dados nos compartilhamentos (Seção 4.1); caracterizamos os seus padrões específicos (Seção 4.2); e apresentamos o nosso gerador de cargas sintéticas (Seção 4.3).

Tabela 2. Estimativas dos volumes de *upload* e *download* (GB) para todas as pastas e para as pastas compartilhadas localmente para diferentes limiares.

	limiar de 10 seg.	limiar de 60 seg.	limiar de 600 seg.
Upload	871,07 (43%)	1201,88 (59,5%)	1518,08 (75%)
Download	942,12 (21%)	1333,17 (29,7%)	2021,59 (45%)
Upload (compartilhadas)	85,59 (9,8%)	117,26 (9,7%)	142,73 (9,4%)
Download (compartilhadas)	222,24 (23,6%)	326,23 (24,5%)	493,45(24,4%)

4.1. Compartilhamento de Conteúdo por Usuários do Campus Universitário

Conforme mostrado na Tabela 1 (Seção 3), 53% dos usuários do campus participam de pelo menos uma pasta compartilhada. Mais ainda, aproximadamente 20% das pastas são compartilhadas por pelo menos dois usuários do campus. A Tabela 2 mostra os volumes de dados transferidos na rede para todas as pastas observadas. Esses números foram computados após realizar as associações de transferências de dados às notificações, seguindo o limiar de 10 segundos (Seção 3). Adicionalmente, também incluimos resultados para limiares de 60 e 600 segundos visando analisar o descarte de dados.

Ao lado de cada volume, indicamos a sua porcentagem em relação a um volume total de referência. As referências para os volumes de *upload* e *download* estimados para todas as pastas (linhas 2 e 3 da tabela) são os volumes totais correspondentes observados nos traços *antes* da associação de transferências de dados para notificações. Essas porcentagens nos permitem avaliar o impacto do limiar no descarte de dados. A referência para os volumes de *upload* e *download* estimados para as pastas compartilhadas (linhas 4 e 5 da tabela) são os volumes estimados nas linhas 2 e 3 da Tabela 2 respectivamente.

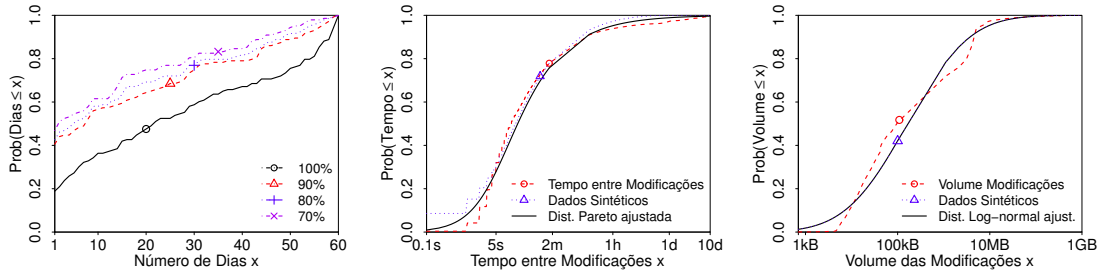
A Tabela 2 mostra que limiares maiores que 10 segundos aumentam a porcentagem de transferências associadas a notificações para todas as pastas (i.e., menor descarte). No entanto, o foco deste trabalho está principalmente no volume de *download* para pastas compartilhadas. Observamos que o valor de limiar tem impacto mínimo na *porcentagem* de *downloads* para pastas compartilhadas, que se mantém em torno de 24% para o período da coleta. Esse volume de dados pode representar um desperdício de banda, já que grande parte desse conteúdo poderia estar replicado localmente no campus. Assim, tão logo um usuário fosse realizar uma atualização, ele o faria sem recorrer aos servidores do Dropbox.

O recurso de LAN Sync, adotado pelo Dropbox para sincronizar arquivos entre dispositivos e pastas compartilhadas na mesma rede local, deveria tomar proveito do grande volume de pastas compartilhadas no campus. Entretanto, a maioria das pastas compartilhadas na universidade contém usuários em sub-redes diferentes. De fato, do total de 3233 pastas compartilhadas (Tabela 1), apenas 18% delas têm todos seus usuários na mesma sub-rede[4].

4.2. Caracterização dos Padrões de Compartilhamento

Para entender os padrões de compartilhamento de pastas no Dropbox, nós primeiro analisamos o tempo em que uma pasta permanece *ativa* no sistema. Intuitivamente, uma pasta é criada e possivelmente compartilhada com um ou mais usuários. Esses usuários

[4]A rede da universidade é tipicamente dividida em sub-redes definidas por prefixos $/24$.



(a) (b) (c)

Figura 1. Modificações em Pastas: (a) Tempo até X% das modificações (X= 70 – 100%); (b) Tempo entre modificações: distribuição Pareto com α=0,460,κ=4,832 (em seg.); (c) Volume de dados por modificação: distribuição LogNormal com μ =5,091 e σ=2,442 (em kB).

realizam várias modificações no conteúdo da pasta nos dias iniciais de sua existência. Com o passar do tempo, esse interesse diminui, até que a pasta torna-se inativa, não recebendo mais modificações. A Figura 1(a) apresenta a distribuição acumulada do número de dias desde a criação de uma pasta até que a mesma tenha recebido $X\%$ de todas as modificações observadas em nossos dados, para X igual a 70%, 80%, 90% e 100%. Essas distribuições foram computadas considerando apenas pastas observadas nos traços da rede por pelo menos 60 dias. Note que cerca de 16% das pastas têm todas as modificações observadas em apenas um dia, enquanto praticamente 50% das pastas recebem todas as modificações em um período inferior a 30 dias. Estes números indicam que muitas pastas recebem rajadas de modificações em curtos períodos de tempo. De fato, a figura mostra que cerca de 45% das pastas recebem 70% de todas as suas modificações apenas no primeiro dia de existência.

A seguir, nós caracterizamos os padrões de modificações em pastas compartilhadas, focando em: número de usuários locais associados a cada pasta, tempo entre modificações sucessivas de uma mesma pasta e volume de dados transferidos por modificação. Para cada variável analisada, nós apresentamos a distribuição estatística que melhor se adequa aos dados, escolhida entre vários modelos usados na literatura. Para cada 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, a distribuição com menor distância de Kolmogorov-Smirnov (contínua) ou menor erro quadrático mínimo (LSE) (discreta) em relação aos dados foi escolhida [Venables and Ripley 2002].

Como mostrado na Tabela 1, cerca de 20% das pastas são compartilhadas por múltiplos usuários locais. Dessas pastas, 13% são compartilhadas por 2 usuários, enquanto na média, o número de usuários locais que compartilham uma mesma pasta é 2,8, e menos de 2% das pastas têm mais de 5 usuários associadas a elas. A distribuição do número de usuários locais associados a cada pasta, omitida por restrições de espaço, foi bem aproximada por uma distribuição Binomial Negativa[5] com parâmetros r=0.225 e p=0.384 e adicionando 1 a cada número aleatório gerado por ela.

[5]Distribuição de probabilidade de massa (PMF) da Binomial Negativa: $p_X(x) = \frac{\Gamma(x+r)}{\Gamma(r)x!}p^r(1-p)^x$.

As pastas armazenadas no Dropbox tendem a sofrer modificações consecutivas em curtos intervalos de tempo. A Figura 1(b) mostra a distribuição acumulada computada sobre todos os intervalos entre modificações em uma mesma pasta, para todas as pastas. Mais de 70% das modificações ocorrem em intervalos inferiores a 1 minuto, embora alguns intervalos superem 1 dia. Esta distribuição enviesada para valores pequenos, que, conforme mostrado na figura, é bem ajustada por uma distribuição Pareto[6], reflete as rajadas de modificações mencionadas anteriormente.

A última variável analisada é o volume de dados associado a cada modificação, cuja distribuição acumulada, computada sobre todas as modificações de todas as pastas, é mostrada na Figura 1(c). Observa-se que as modificações tendem a gerar um tráfego pequeno: cerca de 40% das modificações correspondem a menos de 100 kB, enquanto, na média, cada modificação corresponde a 3 MB. Uma distribuição Log-Normal[7] modela bem o tamanho das modificações de pastas compartilhadas. As curvas "dados sintéticos", nas Figuras 1(b) e 1(c), serão discutidas na próxima seção.

4.3. Gerador de Cargas Sintéticas

Nós desenvolvemos um gerador de cargas sintéticas baseado nos eventos de modificações em pastas caracterizados na Seção 4.2. Esse gerador tem por objetivo produzir um traço de modificações em um dado número n de pastas do Dropbox compartilhadas pelos usuários de um domínio durante um intervalo de tempo ($0 \cdots t_{max}$). Com esse traço, ele visa capturar os volumes de dados transmitidos na rede ao longo do tempo à medida em que um usuário modifica uma pasta (*upload*) e os demais usuários locais que compartilham a mesma pasta obtêem essa modificação (*download*).

Para tal, o gerador modela vários componentes de carga utilizando as distribuições estatísticas que melhor se adequam a eles em dados *reais*. Em particular, ele utiliza as distribuições do número de usuários locais por pasta, tempo entre modificações de uma pasta e volume de dados por modificação discutidos na Seção 4.2. Note que usuários que compartilham uma mesma pasta podem ou não estar conectados ao Dropbox (*online*) quando uma modificação é feita. Assim, como em [Gonçalves et al. 2014], modelamos o comportamento dinâmico de cada usuário do Dropbox a partir das distribuições de duração de sessão (tempo *online*) e tempo entre sessões. Adotamos distribuições Weibull e LogNormal para essas duas variáveis, como apresentado em [Gonçalves et al. 2014].

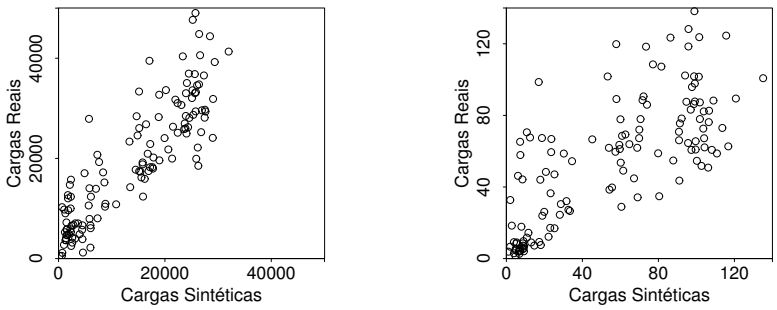
O funcionamento do gerador consiste em, inicialmente, associar a cada pasta os usuários (locais) que a compartilham. Isto é feito utilizando a distribuição Binomial Negativa caracterizada na seção anterior para gerar um número de usuários para cada pasta[8]. Um número de usuários igual a 1 implica que a pasta não é compartilhada com nenhum outro usuário local, embora possa ser compartilhada com usuários externos.

Para cada usuário, o gerador simula sucessivas sessões utilizando as distribuição de duração de sessão e tempo entre sessões. Ele também simula, para cada pasta, uma

[6]PDF da distribuição Pareto: $p_X(x) = \frac{\alpha\kappa^\alpha}{(x+\kappa)^{\alpha+1}}$. Note que esta é uma distribuição Pareto tipo II.

[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]Esta estratégia trata os usuários de cada pasta de forma independente. Alternativamente, também consideramos extrair os usuários associados a cada pasta dos traços reais, preservando assim o número de pastas por usuário. Embora menos realista, a abordagem adotada é mais simples e geral. Além disto, não observamos diferenças significativas entre os traços sintéticos produzidos pelas duas abordagens.



(a) Modificações por dia (b) Volume (GB) por dia

Figura 2. Validação do Gerador de Cargas Sintéticas.

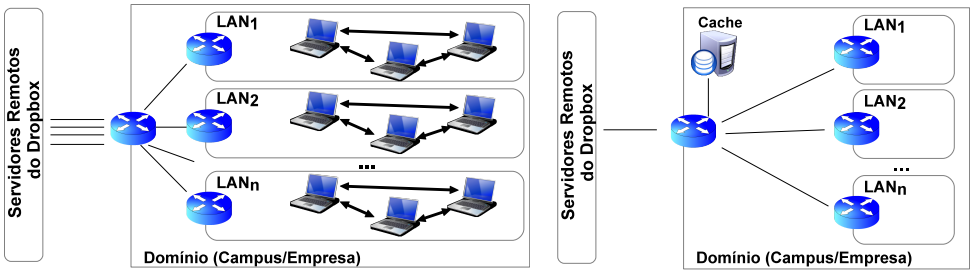
sequência de modificações por usuários locais utilizando as distribuições dos tempos entre modificações e de volume de dados por modificação. Para cada modificação, o gerador contabiliza os volume de *upload* e, no caso de pastas compartilhadas por múltiplos usuários, escalona operações de *download* para cada usuário associado à pasta (exceto o que gerou a modificação). Caso o usuário esteja com uma sessão aberta no instante em que a modificação foi feita, o *download* é feito imediatamente. Caso contrário, ele é escalonado assim que o usuário iniciar uma nova sessão.

Como o foco do gerador é no cálculo do volume agregado de *download* e *upload* por pasta, a escolha do usuário que faz cada modificação é menos importante do que *quando* a modificação é feita. Assim, para fins de geração das cargas sintéticas, nós consideramos que toda pasta será modificada por algum usuário (escritor) com sessão aberta no instante da modificação. Como trabalho futuro, consideramos estender o gerador para capturar o tráfego sob a perspectiva de cada usuário. Neste caso, a escolha do usuário para cada operação (*upload* e *download*) deve ser modelada de forma mais detalhada.

Nós validamos o gerador de duas maneiras. Primeiramente, nós avaliamos que as distribuições das variáveis modeladas extraídas dos traços sintéticos se aproximam das distribuições correspondentes dos dados reais. As distribuições dos dados sintéticos para o tempo entre as modificações sucessivas de uma mesma pasta e o volume dessas modificações são mostradas ns Figuras 1(b) e 1(c) (Seção 4.2): observamos que as distribuições de dados reais e sintéticos são bastante similares. Resultados semelhantes foram obtidos para o número de usuários por pasta. Esta validação pode ser considerada um teste de sanidade já que as distribuições analisadas são entradas do gerador.

Uma validação mais interessante foca em métricas indiretas, que não são explicitamente modeladas pelo gerador. Para realizar tal validação, considerou-se o número e o volume (*upload* e *download*) de modificações ocorridas diariamente nos traços coletados. O gerador produziu um traço sintético para cada dia d utilizando as mesmas distribuições mencionadas acima, mas tendo como alvo o número de pastas observados no traço real naquele dia. Assim podemos comparar os valores das duas métricas em traços reais e sintéticos para cada dia.

Esses valores são mostrados nas Figuras 2(a) e 2(b), onde cada ponto representa um dia. Idealmente, o gráfico deveria apresentar os pontos próximos à reta de 45^o. Entretanto, isto é muito difícil de se alcançar, principalmente quando o processo sendo mode-



(a) Arquitetura Atual (b) Nova Arquitetura

Figura 3. Arquiteturas de Sincronização do Dropbox.

lado é complexo, como no caso dos compartilhamentos no Dropbox. Entretanto, pode-se verificar que o gerador é capaz de capturar as principais tendências dos dados reais já que as correlações lineares entre dados sintéticos e reais são bem altas: 0,87 e 0,74 para número e volume de modificações respectivamente.

A Figura 2(a) mostra que o gerador subestima o número de modificações em muitos dias possivelmente devido aos casos em que as modificações das pastas ocorrem quando usuários associados a ela não estão com sessões abertas. Uma modelagem mais precisa desse processo, incluindo a rede de compartilhamentos [Costa et al. 2014], é bastante desafiadora, sendo deixada para trabalhos futuros. A despeito desta imprecisão, a Figura 2(b) mostra que o gerador tende a superestimar o volume gerado, o que é razoável uma vez que, para o planejamento de capacidade dos servidores de armazenamento ou da capacidade da rede, tais superestimativas levam a decisões mais conservadoras.

Em suma, apesar das imprecisões, consideramos que o gerador captura de forma bastante razoável os padrões de compartilhamento de pastas, principalmente considerando a complexidade do processo modelado. Ele é uma ferramenta bastante útil para suportar avaliações de modificações no protocolo do Dropbox, como a discutida na próxima seção.

5. Nova Arquitetura de Sincronização

Esta seção apresenta e avalia a nossa proposta de modificação na arquitetura de sincronização do Dropbox para reduzir o desperdício de banda devido a *downloads* nos compartilhamentos por usuários locais a um mesmo domínio.

5.1. Descrição da Arquitetura Proposta

A Figura 3(a) apresenta a arquitetura atual do Dropbox, na qual as modificações nas pastas são armazenadas nos servidores remotos. A figura também ilustra o uso de LAN Sync, que permite que modificações de pastas compartilhadas por usuários de uma mesma rede local sejam feitas diretamente entre eles. A nova arquitetura proposta na Figura 3(b) considera a inclusão de um *cache* para armazenar dentro de cada domínio de rede (e.g., um campus ou uma empresa) as modificações de pastas de usuários daquele domínio.

Nossa proposta pressupõe que o Dropbox forneça o serviço especial de *cache* para universidades ou grandes empresas onde muitos usuários utilizam o cliente Dropbox. A universidade ou empresa interessada na implementação do serviço deve fornecer a infraestrutura necessária (i.e., um servidor) para a instalação e funcionamento do mesmo.

O serviço de *cache* funcionaria da seguinte forma. Qualquer modificação feita por um usuário local é armazenada primeiramente no *cache* e posteriormente enviada aos servidores de armazenamento e controle do Dropbox (vide Seção 3). Em seguida, o Dropbox notifica essa modificação para todos os clientes de usuários que compartilham a pasta. Ao receber uma notificação, o cliente primeiramente busca a modificação no *cache* local. Se a modificação estiver armazenada no *cache*, ocorre um acerto (*hit*), caso contrário uma falta (*miss*). Na ocorrência de uma falta, o serviço de cache busca a modificação no Dropbox e a armazena no *cache* para servi-la ao cliente requisitante e a outras possíveis futuras requisições. Contemplamos duas configurações de *cache* em função de quais modificações são armazenadas nele: todas modificações e somente modificações em pastas compartilhadas por múltiplos usuários locais. A primeira configuração é bem mais simples de ser implementada, pois todas as modificações são tratadas indistintamente. A segunda configuração exige uma implementação mais sofisticada, mas pode reduzir a necessidade de espaço de *cache*. Como política de substituição de modificações no *cache*, usamos LRU *Least Recently Used*[9] que é uma política considerada eficiente e frequentemente usada em vários tipos de *cache*. Note que cada modificação de pasta pode ser identificada por um número de versão, o que permite o gerenciamento adequado do conteúdo no *cache*.

5.2. Avaliação

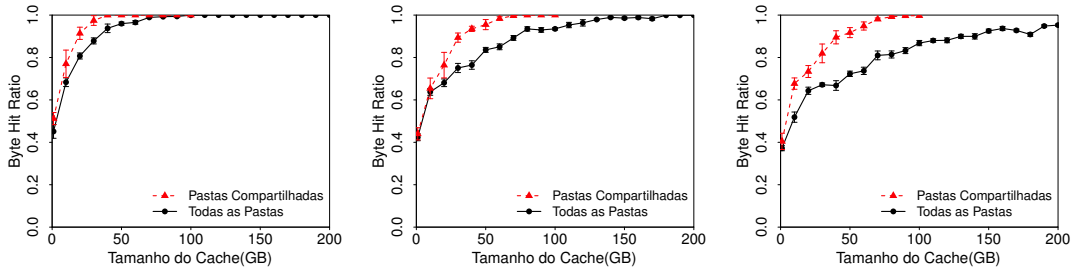
Os benefícios da arquitetura proposta foram avaliados via simulação em cenários construídos utilizando o nosso gerador de cargas sintéticas. Esta avaliação considerou tanto a redução no tráfego de *download* de modificações de pastas compartilhadas por usuários do domínio alvo, quanto o custo do *cache*. A redução no tráfego foi quantificada pelo volume de dados transferidos do cache (*byte hit ratio*), enquanto o custo foi estimado pelo tamanho do *cache*. Note que, conforme discutido na Seção 4.1, a redução máxima de tráfego para os nossos dados, alcançada quando o *byte hit ratio* é 100%, está limitada a 24% do tráfego total de *downloads* (i.e., *downloads* de pastas compartilhadas).

A Figura 4 mostra o *byte hit ratio* para vários tamanhos do cache em três cenários: (1) uma carga de 1065 pastas modificadas por dia, o que corresponde à carga média observada em nossos traços, (2) uma carga de 2185 pastas, que representa o maior número de pastas observadas diariamente em nossos traços, e (3) uma carga bem mais alta com 4000 pastas. Para cada um desses cenários, avaliamos as duas configurações de *cache* mencionadas na seção anterior. A curva preta contínua mostra os resultados quando o *cache* armazena todas as modificações de pastas feitas, independentemente do número de usuários locais que compartilham a pasta[10]. A curva vermelha tracejada mostra os resultados quando o *cache* armazena apenas modificações de pastas compartilhadas por múltiplos usuários locais. Cada resultado na figura corresponde a uma média de 10 replicações, com o intervalo de 95% de confiança correspondente.

Note que um *cache* com apenas 1GB leva a um *byte hit ratio* de 40% nos 3 cenários. Mais ainda, considerando a configuração de *cache* mais simples que armazena todas as modificações, um *cache* com 100GB resulta em um *byte hit ratio* superior a 93%

[9] A avaliação de diferentes políticas está fora do escopo deste trabalho.

[10] Dada a operação do gerador de cargas, somente modificações feitas por usuários locais ao domínio são incluídas nas cargas sintéticas avaliadas. Em um cenário real, modificações feitas por usuários externos também poderiam ser armazenadas no *cache*.



(a) Carga Média (b) Carga Alta (c) Carga Extrema

Figura 4. Benefícios da Arquitetura de *Cache* Proposta: armazenando modificações de todas as pastas ou apenas pastas compartilhadas localmente.

nos cenários de carga média e alta e igual a 87% no cenário com carga mais alta. Isto corresponde a quase o máximo possível de redução de *downloads* dos servidores remotos do Dropbox, particularmente nos dois primeiros cenários. Também podemos notar que o aumento no *byte hit ratio* (e consequente redução de tráfego) para tamanhos de *cache* maiores que 150 GB é próximo de zero para os três cenários. Logo, o uso de *caches* maiores que esse tamanho traz poucos benefícios para domínios cujos padrões de tráfego Dropbox seguem os cenários analisados.

A segunda configuração de *cache* avaliada, que armazena somente modificações de pastas compartilhadas por múltiplos usuários locais, é ainda mais eficiente: um *cache* com somente 50 GB é suficiente para atingir *byte hit ratios* superiores a 95% nos dois primeiros cenários, e igual a 90% no terceiro cenário. Entretanto, ela tem uma complexidade de implementação maior, uma vez que as modificações de pastas devem ser tratadas de forma diferente, dependendo se a pasta é compartilhada localmente ou não. Isto pode refletir em um número maior de mensagens de controle entre servidores remotos, *cache* e usuários. Esta maior complexidade pode não compensar a redução de espaço alcançada, principalmente considerando o baixo custo de memória atualmente.

Em suma, observamos que a arquitetura proposta, mesmo com um *cache* razoavelmente pequeno, resulta em uma considerável diminuição do tráfego entre o domínio analisado (campus universitário) e o Dropbox. Esta redução beneficia não somente o Dropbox mas também os usuários finais, que potencialmente têm uma qualidade de serviço superior dado que o retardo para obtenção dos dados poderá ser bem menor. Mais ainda, ela beneficia a Internet como um todo, com a redução do tráfego nos pontos centrais da rede.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou a primeira caracterização dos padrões de compartilhamento no Dropbox entre usuários de um mesmo domínio, um campus universitário. Esta análise revelou que existe um potencial de economia de banda significativa com a redução de tráfego de dados repetidos e motivou a proposição de uma modificação na arquitetura do Dropbox que inclui *caches* para armazenar modificações nas pastas compartilhadas localmente. Os resultados da caracterização foram usados no desenvolvimento de um gerador de cargas sintéticas que permite a avaliação de modificações no protocolo do Dropbox. Em particular, usamos o gerador para avaliar os benefícios da nova arquitetura proposta em diferentes cenários, concluindo que mesmo *caches* pequenos (e.g., 100GB) permitem um economia de banda bem próxima da máxima possível.

Trabalhos futuros incluem refinamentos no gerador para incluir outros aspectos (e.g., a rede de compartilhamento entre usuários e a criação de novas pastas) e a análise dos custos para implantação de arquiteturas alternativas, como a que foi proposta neste trabalho, que contribuam para a economia de banda.

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. PQS*.

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

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 *Anais do WP2P+*.

Drago, I., Bocchi, E., Mellia, M., Slatman, H., and Pras, A. (2013). Benchmarking Personal Cloud Storage. In *Proc. IMC*.

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

Duarte, R., Cunha, I., Almeida, J., and Vieira, A. B. (2015). Impact of Provider Failures on the Traffic at a University Campus. In *Proc. IFIP Network*.

Finamore, A., Mellia, M., Meo, M., Munafò, M. M., and Rossi, D. (2011). Experiences of Internet Traffic Monitoring with Tstat. *IEEE Network*, 25(3):8–14.

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 *Anais do SBRC*.

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

Hu, W., Yang, T., and Matthews, J. N. (2010). The Good, the Bad and the Ugly of Consumer Cloud Storage. *SIGOPS Oper. Syst. Rev.*, 44(3):110–115.

Li, A., Yang, X., Kandula, S., and Zhang, M. (2010). CloudCmp: Comparing Public Cloud Providers. In *Proc. IMC*.

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. IMC*.

Liu, S., Huang, X., Fu, H., and Yang, G. (2013). Understanding Data Characteristics and Access Patterns in a Cloud Storage System. In *Proc. CCGrid*.

Mager, T., Biersack, E., and Michiardi, P. (2012). A Measurement Study of the Wuala On-line Storage Service. In *Proc. P2P*.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistic with S*. Springer.

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. IWQoS*.

Controlador Resiliente com Distribuição Eficiente para Redes Definidas por *Software*

**Diogo M. F. Mattos, Martin Andreoni Lopez,
Lyno Henrique G. Ferraz e Otto Carlos M. B. Duarte**

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

Resumo. *A distribuição do controle em Redes Definidas por Software melhora a segurança, o desempenho e a escalabilidade da rede. Contudo, essa solução carece de consistência da base de dados e cria hierarquias de controle. Este artigo propõe uma arquitetura eficiente de distribuição de controladores. As contribuições principais da arquitetura proposta são: i) um controlador distribuído em zonas para garantir segurança, desempenho e escalabilidade da rede; ii) uma base de dados única mantida por um controlador designado para prover consistência ao plano de controle; iii) uma heurística de localização otimizada dos controladores para reduzir a latência no plano de controle; iv) um mecanismo resiliente de escolha de controlador designado para assegurar o funcionamento da rede quando há falhas. Um protótipo da proposta foi implementado e a heurística de localização foi analisada em diversas topologias reais. Os resultados mostram que a conectividade é mantida, mesmo em cenários em que há alta taxa de falhas em nós da rede. Por fim, é comprovado que a otimização da localização reduz a latência média dos controladores.*

Abstract. *Control plane distribution on Software Defined Networking enhances security, performance and scalability of the network. In this paper, we propose an efficient architecture for distribution of controllers. The main contributions of the proposed architecture are: i) A distributed controller in zones to ensure security, performance and scalability of the network; ii) A single database maintained by a designated controller to provide consistency to the control plane; iii) An optimized heuristic for locating controllers to reduce latency in the control plan; iv) A resilient mechanism of choosing the designated controller to ensure the proper functioning of the network, even when there are failures. A prototype of the proposal was implemented and the location heuristic was analyzed in real topologies. The results show that connectivity is maintained even in scenarios where there is a high failure rate in network nodes. Finally, it is shown that the optimization of location reduces the average latency of controllers.*

1. Introdução

Redes Definidas por *Software* (*Software Defined Networking* - SDN) advogam pela centralização lógica do plano de controle da rede [Kobayashi et al. 2014, Moraes et al. 2014]. A centralização do controle e a visão global consistente da rede

Este trabalho foi realizado com recursos da CNPq, CAPES, FINEP, FUNTTEL e FAPERJ.

permitem que operadores inovem e implementem novos serviços no núcleo da rede de forma ágil. No entanto, a centralização física do controle em Redes Definidas por *Software* implica desafios para a segurança, o desempenho e a escalabilidade da rede [Kobayashi et al. 2014]. A escalabilidade é prejudicada tanto no diâmetro da rede, comutadores distantes podem não estar próximos de um controlador, assim como no número de comutadores na rede, já que o número de controladores pode não ser suficiente para responder em tempo hábil às requisições de todos comutadores. A centralização física do controle gera um ponto único de falha. O controlador pode ser alvo de ataques de negação de serviço (DoS) [Lopez and Duarte 2015]. Os ataques DoS podem exaurir tanto recursos de processamento, exigindo o atendimento a uma quantidade anômala de requisições pelo controlador, como também pode ocorrer através da poluição (*jamming*) do canal entre controlador e comutadores [Mattos and Duarte 2014].

A distribuição do controle e a replicação de controladores são as principais técnicas para aumentar a resiliência das Redes Definidas por *Software* [Muller et al. 2014]. No entanto, a adoção de controladores que agem como um sistema distribuído [Berde et al. 2014, Koponen et al. 2010] ou a simples replicação de controladores [Kreutz et al. 2013] não é suficiente para garantir a segurança, o desempenho e a escalabilidade, pois dependem ainda do mapeamento otimizado entre comutadores e controladores, criando domínios de controle. Outro desafio que desponta da distribuição do controle é manutenção da consistência da visão global unificada [Levin et al. 2012, Schmid and Suomela 2013]. A visão global inconsistente acarreta em perda de desempenho e erros na execução de aplicações de controle da rede [Schmid and Suomela 2013]. Assim, a distribuição do controle em SDN pode ser entendida sob duas vertentes. A primeira é a provisão de um controlador distribuído com visão global da rede consistente [Koponen et al. 2010, Berde et al. 2014]. A segunda é a otimização da localização e da quantidade de controladores necessários para uma determinada topologia de rede [Heller et al. 2012, Bari et al. 2013].

Este artigo propõe uma arquitetura eficiente para a distribuição de controladores em uma Rede Definida por *Software*. Para tanto, o artigo aborda a distribuição do controle sob a ótica das duas vertentes. Primeiro, desenvolve-se um controlador distribuído para SDN. A proposta consiste em criar zonas de controle independentes com um controlador responsável. Os controladores das zonas se reportam a um controlador designado responsável por manter a consistência da visão global conhecida pelos controladores. Após, modela-se o problema da localização de controladores. O problema é modelado considerando a minimização da latência entre controlador e comutadores e, também, a maximização da conectividade entre controladores. A minimização da latência é um critério importante para diminuir o tempo de configuração de um novo fluxo. A maximização da conectividade entre controladores é importante para garantir-se a resiliência da rede. Um protótipo do controlador proposto foi implementado. A otimização da localização de controladores foi analisada em diferentes topologias. A proposta ainda cria um mecanismo de controladores de salvaguarda (*backup*). No caso de queda de um controlador designado, outro controlador assume. A lista de controladores de salvaguarda é ordenada pelas latências médias dos controladores de zona em relação aos demais.

A distribuição do controle entre diferentes nós físicos, mantendo a visão global de rede centralizada, é um dos principais desafios em SDN [Koponen et al. 2010,

Berde et al. 2014]. Algumas propostas apontam os problemas das Redes Definidas por *Software* e definem soluções iniciais para a distribuição de controle e para a resiliência da rede [Heller et al. 2012, Kreutz et al. 2013]. Outras propostas buscam criar novos controladores distribuídos do ponto de vista de implementação de um sistema operacional distribuído para controlar uma SDN, mas não otimizam a localização dos controladores [Tootoonchian and Ganjali 2010, Koponen et al. 2010, Berde et al. 2014, Hassas Yeganeh and Ganjali 2012]. Há ainda estudos que focam na otimização da localização dos controladores seguindo diferentes heurísticas e objetivos [Muller et al. 2014, Bari et al. 2013]. Essas propostas não visam criar uma arquitetura única que considere desde a concepção do controlador até a localização otimizada dos controladores. Por sua vez, este artigo propõe a arquitetura de controle distribuído com a localização otimizada e mecanismos para garantir a resiliência da rede.

O restante do artigo está organizado da seguinte forma. Os trabalhos relacionados são discutidos na Seção 2. A Seção 3 discute o problema da distribuição e localização de controladores em Redes Definidas por *Software*. O controlador distribuído proposto e a heurísica de localização são apresentados na Seção 4. A Seção 5 apresenta a análise da avaliação do controlador proposto. A Seção 6 conclui o artigo.

2. Trabalhos Relacionados

Levin *et al.* argumentam que a distribuição do controle físico, enquanto o controle lógico mantém-se centralizado, prejudica o desempenho da rede, quando as aplicações de controle centralizado são agnósticas quanto à distribuição do estado [Levin et al. 2012]. Os autores identificam a existência de duas relações de compromisso. A primeira é entre o desempenho das aplicações de controle e a sobrecarga de distribuição de estados entre controladores. A segunda relação é entre a complexidade da lógica das aplicações de controle e a robustez contra inconsistência. Por sua vez, Schmid e Suomela defendem que há aplicações que podem ser executadas por controladores locais, sem que necessitem de uma visão global consolidada [Schmid and Suomela 2013]. Schmid e Suomela identificam dois tipos de planos de controle, o plano de controle horizontal e o plano de controle hierárquico. No plano de controle horizontal cada controlador é responsável por uma área disjunta dos demais e a estrutura se organiza a partir de restrições administrativas. Já no plano de controle hierárquico, os controladores se organizam verticalmente, convergindo para uma visão global da rede conforme se sobe na hierarquia.

A presença de um controlador fisicamente centralizado não está intrinsicamente ligada à arquitetura SDN. Portanto, há propostas de controladores fisicamente distribuídos, mas que mantêm a visão global da rede. A proposta ONIX age como um *middleware* para sistemas distribuídos, propagando as informações para os controladores físicos, e fornece uma API (*Application Programming Interface*) para as aplicações se comunicarem entre as diferentes réplicas e controlarem a rede [Koponen et al. 2010]. Baseado nos mesmos preceitos do controlador ONIX, há também a proposta ONOS [Berde et al. 2014]. O controlador ONOS baseia-se no controlador Floodlight e distribui o estado da rede com o registro distribuído Zookeeper. Outro controlador distribuído é o HyperFlow [Tootoonchian and Ganjali 2010]. Os eventos no HyperFlow são divulgados através do modelo Publicador/Assinante (*Publish/Subscriber*). Tais propostas se baseiam na distribuição do estado entre todos os controladores, mas não preveem uma política de localização e otimização do número de controladores na rede.

Yeganeh *et al.* argumentam que a escalabilidade do controle de Redes Definidas por *Software* pode ser alcançada através de controladores multiprocessados ou através de diversos controladores distribuídos [Yeganeh et al. 2013]. A proposta Kandoo é um controlador SDN hierárquico em que o escopo das aplicações é dividido em local ou global [Hassas Yeganeh and Ganjali 2012]. Aplicações locais e globais coexistem no controle da rede. As aplicações locais são as que podem operar somente com o estado local de cada comutador e, portanto, são implantadas nos controladores mais próximos do plano de dados. Mais próximas dos comutadores, as aplicações podem tratar mais rapidamente as requisições mais frequentes e evitam a sobrecarga no restante da rede de controle. Por sua vez, as aplicações globais são executadas por controladores no topo da hierarquia.

A otimização da localização e o número de controladores para atender uma Rede Definida por *Software* também é um desafio de pesquisa atual. Müller *et al.* propõem uma estratégia de localização de controladores, denominada *Survivor*, que considera a diversidade de caminhos até o controlador, a capacidade dos controladores e os mecanismos de recuperação de falhas [Muller et al. 2014]. A estratégia se baseia em um modelo de Programação Linear Inteira para encontrar a localização ótima dos controladores. O modelo considera que a localização ótima é a que maximiza a conectividade entre controladores e comutadores da rede. Paralelamente, Zhang *et al.* também propõem uma estratégia de localização de controladores para aumentar a resiliência da rede [Zhang et al. 2011]. A ideia central de Zhang *et al.* consiste em minimizar a probabilidade de falha em cada partição de controle da rede. Para tanto, os autores usam um algoritmo de *min-cut* para minimizar duas funções. A primeira minimiza a distância de todos os nós até seu controlador, resultando em uma minimização intra-clusters. A segunda minimiza a conectividade inter-cluster, reduzindo o número de enlaces passíveis de falha entre controlador e comutadores. Bari *et al.* também propõem um esquema de otimização da localização de controladores [Bari et al. 2013].

A proposta deste artigo é um controlador distribuído, com controle horizontal, cuja localização maximize o número de caminhos dos comutadores ao controlador, mas também minimize a latência entre controlador e comutadores de uma partição de controle. Tais funções objetivo são importantes, pois a maximização da quantidade de caminhos entre controlador e comutadores garante resiliência a falhas de enlaces, enquanto a minimização da latência influi diretamente no tempo de configuração de novos fluxos. Assim, a proposta deste artigo age tanto na resiliência quanto na eficiência da rede, enquanto outras propostas atacam apenas uma das vertentes.

3. O Problema da Distribuição e Localização de Controladores

O problema da distribuição e da localização de controladores em SDN é definido por Heller *et al.* como duas características do controle da rede: i) o número de controladores necessários; e ii) o lugar de posicionamento desses controladores. No entanto, essas duas características não exaurem as possibilidades de otimização do controle da rede [Heller et al. 2012]. Elas apenas norteiam a busca por soluções de controle otimizadas. Em adição a esses dois pontos, deve-se considerar uma função objetivo. A função objetivo pode maximizar a distribuição de controladores para tolerar falhas, ao passo que pode também minimizar a latência para aumentar o desempenho e reduzir o tempo de configuração de fluxos.

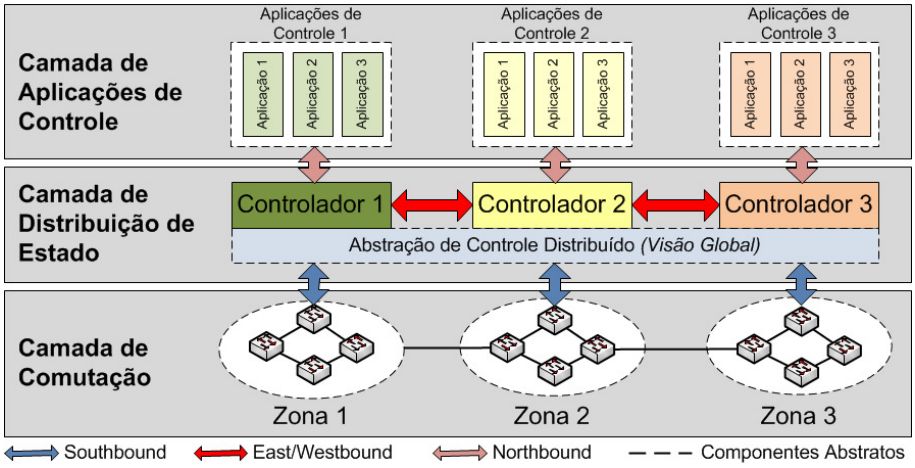


Figura 1. Rede Definida por *Software* com controle distribuído. A rede pode ser dividida em três camadas: comutação, distribuição do controle e aplicações. As setas indicam fluxos de dados de controle na rede. Cada ponto de troca de dados tem uma API (*Southbound, Northbound e East/Westbound*).

Paralelamente ao problema de otimização da localização e da quantidade de controladores, há também o desafio da distribuição do estado na rede de controle. Para tanto, considera-se a arquitetura de rede mostrada na Figura 1. A Figura 1 apresenta os pontos de troca de estado da rede em uma SDN. A ideia é que a rede é divida em três camada lógicas [Schmid and Suomela 2013]. A camada mais baixa, a camada de comutação, é representada pelos elementos de encaminhamento de pacote. Esses elementos armazenam as tabelas de encaminhamento e, no caso de redes OpenFlow [Kobayashi et al. 2014], armazenam também informações relativas aos fluxos e aos contadores de fluxo. A camada de comutação pode ser agrupada em domínios de controle distintos, seja por questões de escalabilidade, seja por questões administrativas.

A camada intermediária, a camada de distribuição de estado de rede, consiste na comunicação entre controladores, divulgando os eventos e o estado local de cada controlador. As propostas ONIX [Koponen et al. 2010] e ONOS [Berde et al. 2014] implementam tal camada de forma semelhante a um *middleware* de sistemas distribuídos. Já o HyperFlow a implementa como o canal de eventos sob o modelo de Publicador/Assinante. Já este artigo propõe a realização da camada de distribuição de estado através da ideia de controlador designado que é responsável por manter uma cópia atualizada da visão global da rede. A camada de distribuição de estado é normalmente referenciada como Sistema Operacional de Rede ("*Network Operating System*") [Levin et al. 2012], pois é essa camada que faz a mediação entre as aplicações de rede e a camada de comutação, permitindo a abstração da camada de comutação para as aplicações. Por fim, a terceira camada lógica é a camada das aplicações de controle. Cada aplicação se comunica com as demais através da camada de distribuição de estado. As aplicações podem executar em nós distintos ou mesmo em um único nó.

Na Figura 1 é possível ressaltar três pontos sensíveis de troca de informação e distribuição do estado. O primeiro é entre os comutadores e os controladores (sistemas operacionais de rede). A implementação mais comum do conceito de SDN, padroniza esse ponto de troca de informação através da API OpenFlow [Kobayashi et al. 2014].

Esse ponto de troca é comumente chamado de *southbound API*. O segundo ponto de troca de informação é entre controladores na camada de distribuição de estado. Nesse caso ocorre a comunicação horizontal entre controladores. A comunicação horizontal entre controladores é chamada de *East/Westbound API*. Por fim, há ainda o ponto de troca de informação entre o controlador, representado na camada de distribuição de estado, e as aplicações. Esse último ponto é chamado de *Northbound API* [Kreutz et al. 2015].

4. O Sistema Proposto

A ideia central do controlador distribuído proposto é dividir o controle da Rede Definida por *Software* em zonas de controle e gerar uma abstração da comunicação da visão global da rede entre todos os controladores de zona. Para tanto, a proposta considera a criação de uma camada de manutenção da consistência entre controladores. Essa camada distribui a visão global da rede entre todos controladores de zona.

4.1. O Controlador Distribuído

A Figura 2 mostra a arquitetura do controlador proposto. Um dos controladores de zona é escolhido como o Controlador Designado. O Controlador Designado é um controlador como os demais, mas que, a partir do momento da sua escolha, passa a ser o responsável por manter a visão global da rede e a distribuí-la para os demais. Nesse contexto, a visão global da rede é considerada como qualquer informação das aplicações de controle que sejam de interesse de mais de um controlador de zona ou que um controlador de zona deseja disponibilizar para os demais.

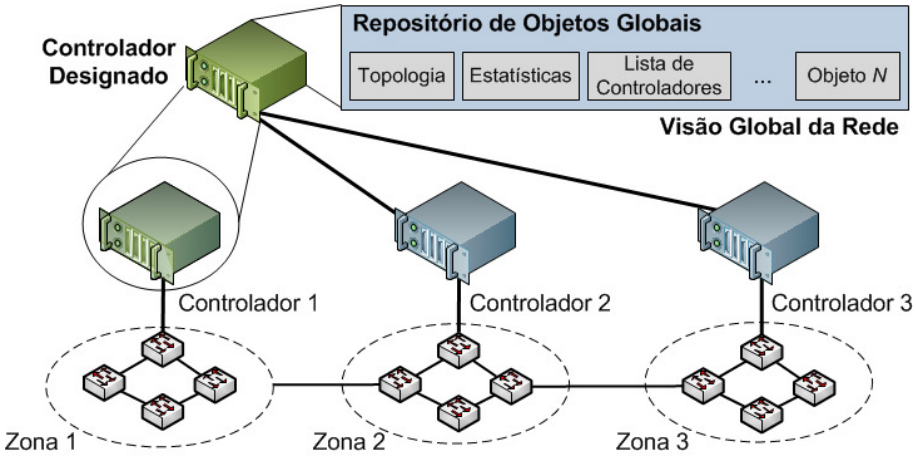


Figura 2. Proposta do controlador distribuído. O Controlador 1 é o Controlador Designado e, portanto, mantém o Repositório de Objetos Globais. O repositório é acessível a todos controladores e permite a distribuição da visão global.

O Controlador Designado armazena e disponibiliza um repositório de objetos de visão global da rede. Um objeto de visão global da rede é uma estrutura de dados que qualquer controlador registra na camada de abstração e distribuição do controle. Quando o objeto é registrado na camada de abstração e distribuição do controle, ele torna-se acessível aos demais controladores, tanto para leitura, quanto para a atualização. O repositório do controlador designado armazena todos objetos registrados.

O registro de objetos na camada de distribuição do controle é importante para abstrair o controle distribuído. Ao acessar um objeto que está registrado no controlador

designado, o controlador de zona pode operá-lo como se fosse local. O controle da consistência do objeto é realizado por parte do controlador designado. Para tanto, um objeto registrado só pode ser atualizado através de um único acesso por vez. As aplicações que executam nos controladores de zona podem realizar o controle local sem que haja a necessidade de se reportarem para o controlador designado, dado que para o controle local não há a necessidade de atualizar a visão global da rede.

No caso de aplicações que atualizam a visão global da rede, como a aplicação de descoberta de topologia, sempre que necessário, cada controlador de zona acessa o objeto referente àquela informação e o atualiza. No caso da descoberta de topologia, o objeto compartilhado é uma coleção de enlaces conhecidos entre os nós e o conjunto dos nós conhecidos da rede. Paralelamente à descoberta de topologia, também pode-se citar o cálculo da árvore de cobertura. Vale notar que, dada a visão global consistente e única da rede, o cálculo da árvore de cobertura pode ser realizado localmente. Cada controlador de zona só é responsável por ativar e desativar a inundação de portas de comutadores da sua zona, sem interferir nos demais [Schmid and Suomela 2013].

A resiliência é alcançada no cenário de controladores distribuídos proposto da seguinte forma. Cada controlador de zona é capaz de assumir o papel de Controlador Designado. Assim, cada controlador de zona possui uma lista ordenada de quais controladores devem assumir o papel de designado em caso de falha. Essa lista é distribuída e mantida ordenada através de um objeto registrado no repositório do controlador designado ativo. Os controladores de zona, de tempos em tempos, consultam a lista ordenada e mantém uma cópia local do estado global da rede, pois no caso de falha do Controlador Designado, o estado global da rede é mantido. A ordenação da lista se dá através da ordem crescente média $L(C_k)$ da latência de um controlador a todos os outros. Ou seja, a cada controlador C_k é associado um valor:

$$L(C_k) = \frac{\sum_{\forall i \in N} Latencia(C_k, C_i)}{N}, \tag{1}$$

onde i é cada nó da rede e N o conjunto de todos os nós da rede, e a lista é ordenada em função do valor $L(C_k)$ de cada controlador. A $Latencia(C_k, C_i)$ na Equação 1 é calculada *a priopri*, de acordo com a topologia da rede, no momento da localização dos controladores. O controlador designado apenas mantém esse valor atualizado de acordo com a monitoração da rede.

4.2. A Localização dos Controladores

A localização dos controladores é definida como um problema de otimização com múltiplos objetivos. Para atingir a máxima resiliência da rede é importante que a localização dos controladores considere que em um cenário de falha, a rede continue conexa e que, caso haja partição da rede, cada partição possua um controlador em seu interior. Paralelamente, é importante que para o funcionamento adequado da rede, o tempo de configuração de um novo fluxo seja o mínimo possível [Heller et al. 2012, Bari et al. 2013]. Assim, é importante que o tempo de comunicação entre o controlador e o conjunto de comutadores controlados seja o mínimo possível. Assim, os objetivos da otimização da localização de controladores são dados por:

$$min \prod_{\forall i \in N} (PercentualRede(C_i) * Prob(C_i))^{y_i}, \quad (2)$$

onde y_i identifica se o nó i é controlador ($y_i = 1$) ou não ($y_i = 0$), $PercentualRede(C_i)$ indica o percentual da rede que fica sem controlador caso haja uma falha em C_i e N representa o conjunto de todos os nós na rede. A equação

$$min \sum_{\forall i \in N} \frac{\sum_{\forall k \in N} y_i * y_{i,k} * latencia(C_k, C_i)}{N} \quad (3)$$

minimiza a latência, em que $y_{i,k}$ representa que o controlador i controla o nó k e y_i representa se o nó de índice i é um controlador ou não. As equações valem para $i, k \in N$, onde N é o conjunto de todos os nós da rede.

A Equação 2 é confusa a primeira vista. Contudo, sua interpretação é fácil. A ideia da equação é minimizar o percentual da rede, representado pela função $0 < PercentualRede(C_i) \leq 1$, que fica sem controle caso o controlador C_i falhe. A função $Prob(C_i)$ modela a probabilidade de falha do nó C_i. No entanto, a falha do nó C_i só é prejudicial para o controle da rede caso C_i seja um controlador. Assim, cada termo do produtório é elevado a y_i que determina se o nó de índice i é controlador ou não. No caso de ser controlador, o termo é considerado. No caso contrário, o termo é elevado a 0, resultando no valor 1, que não interfere no produtório. Assim, ao minimizar a Equação 2, procura-se uma solução que mantenha a rede conectada e com controle, mesmo em cenários de falha. A Equação 3 é de interpretação imediata. Essa equação busca uma solução de localização que minimize a latência média entre todos os controladores e os comutadores que ele controla.

4.3. A Heurística de Localização

As Equações 2 e 3 definem a ideia central da estratégia de localização proposta. No entanto, para aplicar a estratégia proposta, desenvolve-se uma heurística de cálculo da localização de controladores otimizada. A heurística se baseia no método de otimização de Arrefecimento Simulado[1] [Cardoso et al. 2014].

O problema da localização foi modelado em Arrefecimento Simulado da seguinte forma. A solução do problema é um vetor com Y posições, onde Y é o número de controladores que se deseja alocar na rede. A cada iteração é gerada uma nova solução candidata que pode ser aceita caso tenha um custo menor do que a anterior. Se a solução não for melhor que a anterior, ela tem uma probabilidade associada a ela. Sorteia-se um número aleatório e verifica-se se o número é menor que a probabilidade de aceitação da solução. Em caso positivo, a solução é aceita mesmo tendo um custo maior do que a anterior. Esse comportamento é necessário para evitar que a otimização convirja para um mínimo local.

[1]O método de otimização de Arrefecimento Simulado foi escolhido como meta-heurística, pois esse método tem a convergência para o mínimo global comprovada, em tempo indeterminado, mesmo no cenário de decaimento linear de temperatura. Para tanto, a perturbação usada para a geração de novas soluções segue distribuição Cauchy.

Algoritmo 1: Mapeamento dos comutadores em controladores pertencentes ao vetor solução. A função $Distancia$ é definida como a latência entre os nós.

```
G = Grafo(topologia_da_rede)
controladoresArray = array de controladores calculados
mapArray = array [tamanho(G)] de inteiro
mapArray[k] = 0 ∀k ∈ G
for i ∈ tamanho(mapArray) do
    for j ∈ tamanho(controladoresArray) do
        if Distancia(controladoresArray[j], G[i]) <
        Distancia(mapArray[i], G[i]) then
            mapArray[i] = controladoresArray[j]
        end
    end
end
return mapArray
```

Dado o vetor solução retornado pela meta-heurística de Arrefecimento Simulado, dois algoritmos são aplicados sobre ele. O Algoritmo 1 mostra o mapeamento dos comutadores nos controladores do vetor solução. Esse algoritmo define as zonas de controle. O Algoritmo 2 mostra o cálculo da função objetivo que é o quanto da rede permanece conectado mesmo em um cenário com probabilidade de falha de nós.

O Algoritmo 2 recebe como entrada o grafo G, contendo toda a topologia da rede, um vetor de probabilidade $prob$, no qual cada posição i representa a probabilidade de falha do nó i da rede, e o vetor de mapeamento de comutadores em controladores. A saída do algoritmo é o inverso do percentual de nós da rede que permanecem conectados e com controle, mesmo no cenário de falhas representado pelo vetor $prob$. É importante ressaltar que o valor retornado é o inverso do percentual de nós que permanecem controlados, pois o problema de localização foi desenvolvido e modelado como um problema de minimização. Sendo assim, a minimização do inverso é a maximização dos nós que permanecem controlados mesmo no cenário de falhas.

5. A Avaliação do Sistema Proposto

A proposta foi avaliada em duas etapas. A primeira etapa foi a implementação de um protótipo do controlador distribuído. O protótipo foi submetido a experimentos para verificar a consistência da visão global da rede e o ganho de desempenho ao distribuir o controle e submeter o sistema a altas taxas de requisição de fluxos/segundo. A segunda etapa de avaliação consiste na otimização da localização de controladores em diferentes topologias reais e, posteriormente, a localização otimizada encontrada é avaliada quanto à latência média entre controladores e comutadores e quanto à resiliência provida à rede. A heurística de otimização proposta é comparada com heurísticas com diferentes funções objetivo: menor latência entre controlador e comutadores, centroides da rede, menor número de saltos entre controlador e comutadores.

O protótipo do controlador proposto, seguindo o modelo de Controlador Designado, considera uma rede com comutadores OpenFlow e implementa o controle dis-

Algoritmo 2: Função objetivo da otimização. Medida da partição da rede em cenário de falha.

```
G = Grafo(topologia_da_rede)
mapArray = mapeamento de cada comutador em um controlador
prob = array com a probabilidade de falha de cada nó em G
for k ∈ G do
    aleatorio = random()
    if aleatorio ≤ prob[k] then
        G.remove(k)
    end
end
componentesConexas = conjunto_componentes_conexas(G)
soma = 0 %soma de todos os nos controlados em componentes conexas
for componente ∈ componentesConexas do
    for no ∈ componente do
        if mapArray[no] ∈ component then
            soma+ = 1
        end
    end
end
return tamanho(G)/soma %inverso do percentual da rede com controle
```

tribuído sobre o controlador POX[2]. O repositório de objetos globais foi implementado usando-se uma versão adaptada do módulo para a programação distribuída em Python DOPY [3]. Assim, o Controlador Designado escolhido, inicia um servidor em que os demais controladores se conectam a ele para manter a consistência do repositório de objetos globais. Vale ressaltar que para realizar os experimentos com o protótipo, as aplicações `forwarding.l2_learning` e `openflow.spanning_tree` do controlador POX foram adaptadas para registrar e consultar o repositório de objetos globais ao invés de manterem a visão de cada controlador somente local.

O protótipo do controlador foi executado como diversos processos em um computador pessoal Intel i7-2600 @ 3.40 GHz , com 16 GB de RAM, executando Debian Linux. Os comutadores OpenFlow são quatro computadores pessoais Core 2 Duo @ 2.40 GHz, com 3 GB de RAM, executando Debian Linux e o comutador por software com suporte a OpenFlow, Open vSwitch [4].

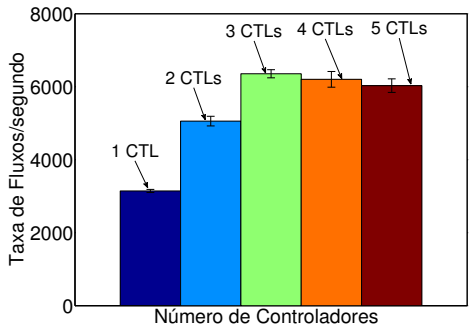
A Figura 3 mostra a avaliação do controlador distribuído proposto. O primeiro experimento visa avaliar o número de fluxos/segundo que o plano de controle consegue atender. Para avaliar o plano de controle, foi usada a ferramenta `cbench`[5]. `Cbench` emula diversos comutadores conectados a um controlador e gera eventos de `packet_in` para medir a capacidade do controlador a reagir a esses eventos. A Figura 3(a) evindencia que ao adicionar mais controladores ao plano de controle, a taxa de fluxos/segundo

[2]http://www.noxrepo.org/pox

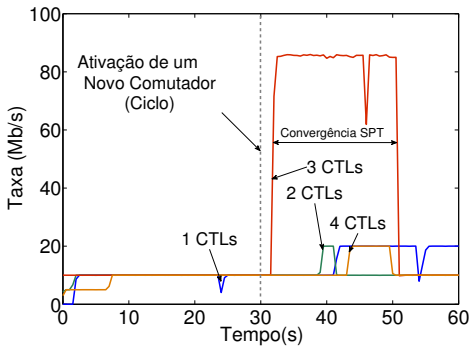
[3]http://www.mindhog.net/ mmuller/projects/dopy/.

[4]http://openvswitch.org/.

[5]http://www.openflowhub.org/display/floodlightcontroller/Cbench.



(a) Execução do `cbench` para 1, 2, 3, 4 e 5 controladores (CTLs no gráfico). Para 4 e 5 controladores o número de fluxos/s atendidos é menor do que para 3 controladores devido a limitações de processamento da máquina de teste.



(b) Ativação de um novo comutador na rede, em 30 s, gerando um ciclo. Durante a convergência do algoritmo de árvore de cobertura (SPT) há duplicação de pacotes na rede.

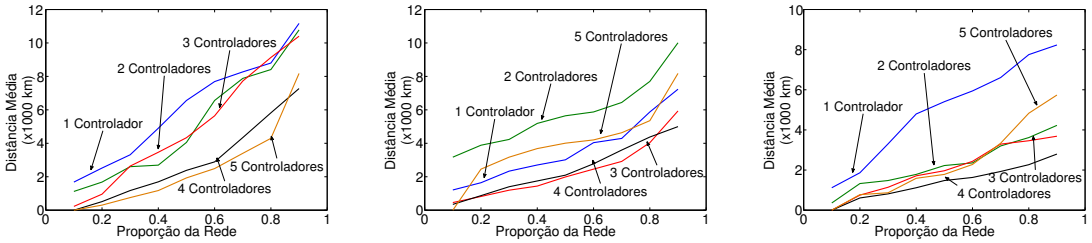
Figura 3. Avaliação do protótipo de controlador distribuído. a) Avaliação quanto ao número agregado de fluxos atendidos pelos controladores na rede. b) Reação do controle distribuído à entrada de um novo comutador na rede.

aumenta. Com três controladores, a taxa/fluxos por segundo atendida é quase o dobro de quando há somente um controlador na rede. Contudo, com 4 ou 5 controladores a taxa de fluxos atendidos é menor do que com três controladores, pois há concorrência entre os processos de controle e geração de fluxo e, portanto, há um gargalo de processamento, já que com 4 controladores os 8 núcleos (`cores`) do computador que hospeda o teste são usados para gerar fluxos e reagir aos eventos de `packet_in`. O segundo experimento verifica a reação do plano de controle à ativação de um novo comutador, fechando uma topologia em anel com os 4 comutadores. O comutador é ativado aos 30 s. Percebe-se que após a ativação há a duplicação de pacotes devido à inconsistência na árvore de cobertura. No pior caso, com três controladores, a convergência da árvore de cobertura chega a demorar 20 s, e converge para que todos os comutadores tenham acesso à visão global.

A avaliação da heurística de localização proposta considera três topologias: topologia da rede da RNP no Brasil, da rede GEANT na Europa e da rede MPLS da AT&T nos Estados Unidos. Os grafos das topologias usadas foram obtidos no *The Internet Topology Zoo*[6]. A rede da RNP conta com 31 nós, a GEANT, com 40 nós e a AT&T conta com 25 nós. A otimização considera que cada nó na rede é capaz de suportar um controlador.

Como o conjunto de dados disponibilizados não apresenta informação sobre latência entre os nós, a avaliação usa a distância física entre os nós como medida indireta da latência. Tal procedimento é aplicado de maneira semelhante por Heller *et al.* [Heller et al. 2012]. A Figura 4 mostra o comportamento da latência em função do percentual de nós na rede. A partir dessa figura, pode-se observar que ao adicionar o quinto controlador, a redução da latência não é tão significativa. No caso da topologia da AT&T, mostrada na Figura 4(c), com 5 controladores ocorre o processo inverso e há um aumento da latência. Portanto, os próximos resultados, por mérito de clareza, consideram somente os cenários com 4 controladores.

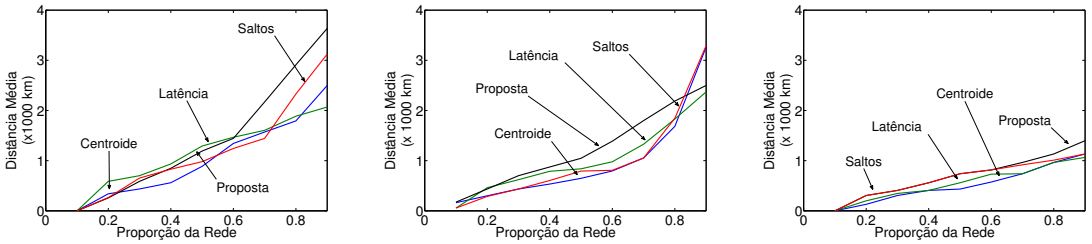
[6]http://www.topology-zoo.org/.



(a) Topologia da RNP no Brasil. (b) Topologia da GEANT na UE. (c) Topologia da AT&T nos EUA.

Figura 4. Latência média entre comutadores e o comutador a que foi mapeado em função do percentual da rede. Heurística proposta de localização considera maior resiliência e menor latência média. Os resultados mostram para cada topologia o uso de até cinco controladores.

Os experimentos seguintes comparam a heurística proposta com três outras: menor latência entre controlador e comutadores (`Latência`), centroides da rede (`Centroide`), menor número de saltos entre controlador e comutadores (`Saltos`). A heurística de menor latência é inspirada na proposta de [Heller et al. 2012]. A heurística da escolha dos centroides da rede é baseada na estratégia de [Zhang et al. 2011]. Por fim, a heurística de escolha dos controladores com o menor número de saltos até os comutadores é uma proposta simplista de comparação com as demais. As heurísticas foram usadas como função objetivo para a otimização do Arrefecimento Simulado.

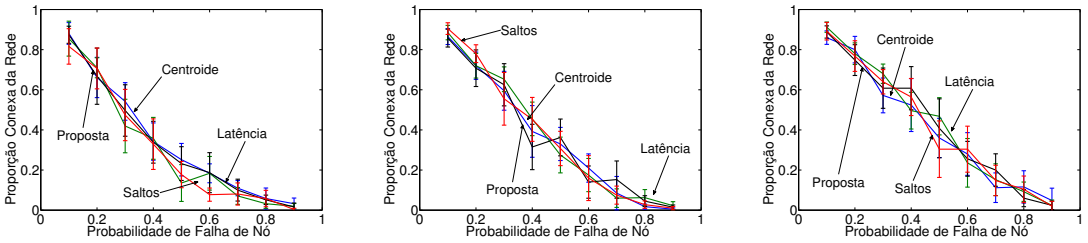


(a) Topologia da RNP no Brasil. (b) Topologia da GEANT na UE. (c) Topologia da AT&T nos EUA.

Figura 5. Latência média entre comutadores e o comutador a que foi mapeado em função do percentual da rede. Comparação entre a `Proposta`, a menor latência (`Latência`), os centroides da rede (`Centroide`) e o menor número de saltos (`Saltos`). Todas as heurísticas consideram a localização de 4 controladores.

A Figura 5 mostra as quatro heurísticas comparadas quanto a latência média entre os controladores e os comutadores. Na três topologias, evidencia-se que a heurística proposta apresenta um desempenho similar às demais. Vale observar, que a heurística do menor número de saltos entre comutador e controlador (`Saltos`), embora seja simplista, apresenta baixa latência média. Esse resultado é compatível com os cenários, pois os nós das topologias consideradas tendem conectarem diretamente com nós próximos.

O último experimento avalia a resiliência da rede, quando a localização dos controladores é realizada através das quatro heurísticas avaliadas. Para tanto, foi desenvolvido um simulador de falhas na topologia. Os resultados da Figura 6 mostram a proporção da rede que continua conexa e controlada em função da probabilidade de falha dos nós



(a) Topologia da RNP no Brasil. (b) Topologia da GEANT na UE. (c) Topologia da AT&T nos EUA.

Figura 6. Proporção da rede que continua conexa e controlada em função da probabilidade de falha dos nós. Comparação entre a `Proposta`, a menor latência (`Latência`), os centroides da rede (`Centroide`) e o menor número de saltos (`Saltos`). Todas as heurísticas consideram a localização de 4 controladores.

da rede. Uma partição da rede é considerada funcional, caso a partição contenha algum nó em que foi implantado um controlador. Essa suposição é feita, pois os comutadores dessa partição podem ter uma lista de controladores de salva-guarda que contenha todos os controladores da rede, ordenados[7].

Os resultados das Figura 6(a) e 6(b) demonstram que a heurística proposta tende a manter uma maior proporção da rede conexa quando a probilidade de falhas dos nós é até 0.5. Contudo, o ganho é limitado quando comparado com outras heurísticas, pois as topologias consideradas não apresentam redundância capaz de suportar falhas de múltiplos nós. Assim, mesmo com uma probabilidade pequena de falha de nós, as partições geradas na rede ficam sem controlador e, portanto, são consideradas como falhas também. Evidencia-se também que a heurística simples de alocação de controladores em nós com menor número médio de saltos até os demais (`Saltos`) apresenta bons resultados de resiliência. A heurística simples `Saltos`, com menor tempo de convergência, é então uma boa opção em topologias com baixo grau de redundância.

6. Conclusão

As Redes Definidas por *Software* (SDN) utilizam a centralização lógica do plano de controle para criar uma abstração da visão global da rede que facilita o desenvolvimento e a implantação de novos serviços no núcleo da rede. Esse artigo propôs uma arquitetura de controladores para SDN que mantém a visão global da rede, ao passo que cria zonas de controle com controle distribuído. A visão global da rede é alcançada através da ideia de Controlador Designado, que é um controlador de zona que assume o papel de manter a consistência da visão global da rede. O controlador designado mantém um repositório de objetos globais que é acessível a todos controladores. Paralelamente ao controlador distribuído, esse artigo também propôs duas heurísticas de otimização da localização dos controladores: uma baseada na maximização da resiliência da rede e, outra, mais simples, baseada na minimização do número de saltos entre controlador e comutadores. Um protótipo do controlador proposto foi implementado e avaliado. Os resultados mostraram que a visão global é mantida no cenário de controle fisicamente distribuído e todos os controladores a acessam e convergem no cálculo de uma árvore de

[7]Considera-se o uso do OpenFlow 1.2, ou superior, para a definição de papéis entre controladores, permitindo a configuração de controladores de salvaguarda.

cobertura comum. As heurísticas de otimização foram avaliadas e os resultados mostraram que a heurística proposta aumenta a resiliência da rede, em especial quando a taxa de falhas é de até 0.5, mas para topologia com menor redundância, a heurística mais simples, o menor número médio de saltos, apresenta desempenho comparável às demais.

7. Referências

[Bari et al. 2013] Bari, M., Roy, A., Chowdhury, S., Zhang, Q., Zhani, M., Ahmed, R., and Boutaba, R. (2013). Dynamic controller provisioning in software defined networks. In *Network and Service Management (CNSM), 2013 9th International Conference on*, pages 18–25.

[Berde et al. 2014] 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.

[Cardoso et al. 2014] Cardoso, L. P., Ferraz, L. H. G., and Duarte, O. C. M. B. (2014). Migração de máquinas virtuais para economia de energia. In *Workshop de Gerência e Operação de Redes e Serviços (WGRS 2014) do SBRC'2014*.

[Hassas Yeganeh and Ganjali 2012] Hassas Yeganeh, S. and Ganjali, Y. (2012). Kandoo: A framework for efficient and scalable offloading of control applications. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 19–24, New York, NY, USA. ACM.

[Heller et al. 2012] Heller, B., Sherwood, R., and McKeown, N. (2012). The controller placement problem. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 7–12, New York, NY, USA. ACM.

[Kobayashi et al. 2014] Kobayashi, M., Seetharaman, S., Parulkar, G., Appenzeller, G., Little, J., Van Reijendam, J., Weissmann, P., and Mckeown, N. (2014). Maturing of openflow and software-defined networking through deployments. *Comput. Netw.*, 61:151–175.

[Koponen et al. 2010] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., and Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA. USENIX Association.

[Kreutz et al. 2013] Kreutz, D., Ramos, F. M., and Verissimo, P. (2013). Towards secure and dependable software-defined networks. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 55–60, New York, NY, USA. ACM.

[Kreutz et al. 2015] Kreutz, D., Ramos, F. M. V., Veríssimo, P., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):63.

[Levin et al. 2012] Levin, D., Wundsam, A., Heller, B., Handigol, N., and Feldmann, A. (2012). Logically centralized?: state distribution trade-offs in software defined networks. In *Proceedings of the First workshop on Hot topics in software defined networks*, HotSDN '12, Helsinki, Finland. ACM.

[Lopez and Duarte 2015] Lopez, M. E. A. and Duarte, O. C. M. B. (2015). Providing elasticity to intrusion detection systems in virtualized software defined networks. In *IEEE ICC 2015 - Communication and Information Systems Security Symposium (ICC'15 (11) CISS)*, London, United Kingdom.

[Mattos and Duarte 2014] Mattos, D. M. F. and Duarte, O. C. M. B. (2014). AuthFlow: Um mecanismo de autenticação e controle de acesso para redes definidas por software. In *XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2014*.

[Moraes et al. 2014] Moraes, I. M., Mattos, D. M., Ferraz, L. H. G., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M., de Amorim, M. D., Velloso, P. B., Duarte, O. C. M., and Pujolle, G. (2014). Fits: A flexible virtual network testbed architecture. *Computer Networks*, (0):–.

[Muller et al. 2014] Muller, L. F., Oliveira, R. R., Luizelli, M. C., Gaspary, L. P., and Barcellos, M. P. (2014). Survivor: an enhanced controller placement strategy for improving sdn survivability. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, Austin, Texas, USA.

[Schmid and Suomela 2013] Schmid, S. and Suomela, J. (2013). Exploiting locality in distributed sdn control. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 121–126, New York, NY, USA. ACM.

[Tootoonchian and Ganjali 2010] Tootoonchian, A. and Ganjali, Y. (2010). Hyperflow: A distributed control plane for openflow. In *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, pages 3–3, Berkeley, CA, USA. USENIX Association.

[Yeganeh et al. 2013] Yeganeh, S., Tootoonchian, A., and Ganjali, Y. (2013). On scalability of software-defined networking. *Communications Magazine, IEEE*, 51(2):136–141.

[Zhang et al. 2011] Zhang, Y., Beheshti, N., and Tatipamula, M. (2011). On resilience of split-architecture networks. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6.

Uma Análise do Custo do Tráfego de *Spam* para Operadores de Rede

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

[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

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

{cristine,jessen,mhp}@cert.br

Abstract. *Spam messages are used to disseminate malware, make phishing attacks, and advertise illegal products. Spam generates costs to users, e.g., victims of phishing, and to network administrators, e.g., who provision and pay for the traffic. Recent proposals aim to identify and filter spam messages at the origin, restraining message propagation and reducing wasted bandwidth on the route from the spammer to the destination. In this work we analyze spam traffic costs for network operators. We measure the routes traversed by real spam messages colected at five honeypots, and estimate spam traffic costs according to the business relationships between networks traversed on each route. We show that stub networks are systematically encumbered by high spam traffic costs but can cooperate to filter up to 70% of spam messages at the origin. Our results also indicate that transit networks that send a lot of spam may employ traffic engineering to reduce their transit costs.*

Resumo. *Mensagens de* spam *são utilizadas para propagação de* malware*, ataques de* phishing *e venda ilegal de produtos. O* spam *gera custos para usuários, e.g., vítimas de* phishing*, e para operadores de rede, e.g., que pagam pela transferência do tráfego. Propostas recentes visam identificar e filtrar mensagens de* spam *na origem, reduzindo o alcance das mensagens e evitando desperdício de banda de rede na rota entre a origem e o destino. Neste trabalho fazemos uma análise do custo do tráfego de* spam *para operadores de rede. Medimos rotas trafegadas por mensagens de* spam *reais coletadas em cinco* honeypots *e estimamos o custo do tráfego de* spam *de acordo com as relações comerciais entre as redes que compõem cada rota. Nossos resultados mostram que algumas redes são sistematicamente oneradas pelo tráfego de* spam *mas que podem cooperar para filtrar até 70% das mensagens de* spam *na origem. Por último, mostramos indícios de que redes que enviam muito* spam *utilizam mecanismos de engenharia de tráfego para reduzir o custo do tráfego.*

*Osvaldo Fonseca e Elverton Fazzion contribuiram de forma equivalente para este trabalho.

1. Introdução

Mensagens de *spam*, e-mails não solicitados enviados para uma grande quantidade de destinatários, estão frequentemente associadas a atividades ilegais. Apesar da maior parte das mensagens de *spam* distribuírem propaganda de produtos ou serviços, *spam* também é utilizado para atrair usuários a réplicas falsas de serviços reais (*phishing*) [Orman 2013] ou propagação de programas maliciosos (*malware*) [Newman et al. 2002]. Mensagens de *spam* corresponderam a 90% do tráfego de mensagens de correio eletrônico em 2013, gerando aproximadamente 216 TB de volume diário [Symantec 2014].

A batalha contra os *spammers* se trava em diversas frentes. Nos últimos anos, muitos trabalhos têm focado em combater o *spam* na origem, de forma a evitar que essas mensagens trafeguem pela Internet da origem até o destino consumindo banda de rede [Las-Casas et al. 2013]. Existem trabalhos na literatura que mostram que o tráfego de *spam* gera um alto custo para a Internet [Sipior et al. 2004, Rao and Reiley 2012]. Entretanto, estes trabalhos consideram apenas o custo agregado e não uma granularidade menor para identificar quais redes são oneradas pelo tráfego de *spam*.

Um Sistema Autônomo (AS) na Internet é uma entidade registrada junto às autoridades da Internet, e.g., ARIN e LACNIC, como operadora de recursos de rede conectados à Internet, como roteadores, enlaces, computadores e faixas de endereços IP. ASes estabelecem relações comerciais com outros ASes, frequentemente envolvendo pagamento por serviços ou por conectividade global. Envio e recebimento de mensagens de *spam* pode resultar em custos diretos para ASes que pagam por tráfego.

Neste trabalho avaliamos o custo do tráfego de *spam*, em bytes, para operadores de rede na granularidade de ASes. Propomos uma abordagem que permite entender quais ASes estão sendo onerados ou beneficiados pelo tráfego de *spam* (seção 2). O primeiro desafio é obter uma amostra representativa do tráfego de *spam* na Internet. Nossa análise é baseada em 78,5 milhões de mensagens de *spam* reais coletadas por *honeypots* instalados em cinco redes em quatro países distintos. O segundo desafio é estimar as rotas trafegadas pelas mensagens de *spam*. Realizamos *traceroutes* de diversos pontos na Internet e utilizamos técnicas de mapeamento de endereços IP em ASes [Chen et al. 2009] para inferir a sequência de ASes nas rotas trafegadas pelas mensagens de *spam*. O terceiro desafio é inferir as relações comerciais entre ASes e a direção do fluxo monetário incorrido pelo tráfego de *spam*. Como os contratos de custos entre ASes são privados, o custo monetário é impossível de obter. Desta forma, utilizamos as relações entre ASes e o tráfego de *spam* em nossa análise. Estas relações foram obtidas de bases dados públicas [Luckie et al. 2013, Giotsas et al. 2014]. Combinando estas diversas fontes de dados conseguimos inferir quanto tráfego de *spam* trafega entre pares de ASes e estimar o custo para cada AS.

Nossos resultados têm aplicações práticas diretas. Identificamos quais ASes são onerados e têm incentivos para filtrar tráfego de *spam* e quais ASes se beneficiam do tráfego de *spam*. Este resultado facilita a cooperação entre redes oneradas para combater o *spam*, e.g., através da instalação de filtros distribuídos de *spam* na origem. Em particular, redes de borda são as mais oneradas e as que têm maior incentivo para a implantação de mecanismos para filtragem e bloqueio de *spam* na origem. Além disso, mostramos que, se algumas redes cooperassem, poder-se-ia filtrar até 70% das mensagens de *spam* na origem. Ainda mais, mostramos o impacto no volume de tráfego que as mensagem

de *spam* podem ter se encaminhadas por servidores SMTP e como a instalação de filtros que impeçam esse fato poderiam reduzir drasticamente o tráfego. Por fim, nossos resultados também motivam novas técnicas para identificação do tráfego de *spam*, e.g., detectar engenharia de tráfego com objetivo de amortizar rajadas e custo do tráfego.

2. Conjuntos de dados

Nesta seção explicamos como capturamos mensagens reais de *spam* representativas do tráfego global (seção 2.1) e como inferimos as rotas trafegadas pelas mensagens na Internet utilizando medições distribuídas a bases de dados diversas (seção 2.2).

2.1. Coleta de mensagens de *spam*

As mensagens de *spam* que utilizamos em nossa análise foram coletadas de cinco *honeypots* instalados em diferentes países, dois no Brasil, um na Holanda, um nos Estados Unidos e um no Uruguai, em redes de diferentes características. *Honeypots* são máquinas que simulam servidores vulneráveis para atrair *spammers*. Os *honeypots* são configurados para *simular proxies* HTTP e SOCKS bem como *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.[1] 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. Estes serviços 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 toda a interação com os *honeypots* provém de *spammers*. Toda interação com os *honeypots* é registrada e as mensagens de *spam* são armazenadas localmente. Nenhuma mensagem de *spam* é efetivamente entregue ao seu destino nem conexões SMTP via proxies efetivamente estabelecidas—exceto para mensagens classificadas como mensagens de teste segundo regras pré-definidas.[2] Periodicamente, ao longo de cada dia, todo o spam armazenado nos *honeypots* é copiado para os servidores centrais do projeto. A distribuição dos *honeypots* em diferentes países e diferentes redes (e.g., redes acadêmicas e comerciais no Brasil) visa obter uma visão geral do tráfego de *spam* na Internet. Neste trabalho analisamos mensagens coletadas entre 01/08/2014 e 31/08/2014. A tabela 1 oferece uma visão geral dos dados coletados. Um detalhe importante sobre os dados utilizados no trabalho é que eles são de natureza sensível e não podem ser publicados, por oferecerem diversas formas para identificação dos honeypots e conterem endereços válidos de usuários que seriam alvo de spam. A sanitização dessas informações reduziria grandemente o interesse dos dados.

Neste período coletamos 78,5 milhões de mensagens, provenientes de 722 ASes de origem distintos em 122 países. O número de endereços IP que utilizam o protocolo SMTP é maior (98,50% do total) do que daqueles que utilizam HTTP/SOCKS e enviam 56,01% das mensagens de *spam*. Estas observações seguem padrões observados em trabalhos anteriores, cujos endereços IP que utilizam SMTP são típicos de participantes em

[1] Servidores SMTP de *relay* aberto encaminham mensagens de e-mail recebidas para outros servidores.

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

Tabela 1. Visão geral da base de *spam*.

MÉTRICA	HTTP/SOCKS	SMTP	TOTAL
Mensagens (milhões)	34,54 (44%)	43,99 (56%)	78,53
Endereços IP de origem	491 (2%)	32 336 (98%)	32 826
Prefixos de rede de origem	264 (18%)	1 243 (83%)	1 492
Sistemas autônomos de origem	110 (15%)	653 (90%)	722
Código de países (IPs de origem)	36 (30%)	118 (97%)	122
Volume de Tráfego (GB)	129,34 (27%)	341,33 (73%)	470,68

Tabela 2. Quantidade de *spam* por *honeypot*.

ID	MILHÕES DE MENSAGENS	ENDEREÇOS IP DE ORIGEM	ASES DE ORIGEM	ASES NÃO COBERTOS	VOLUME (GB)
BR-01	12,25 (16%)	31 084 (95%)	552 (76%)	328 (59%)	91,33 (19%)
BR-02	19,51 (25%)	24 317 (74%)	178 (25%)	82 (46%)	85,01 (18%)
NL-01	22,99 (29%)	23 264 (71%)	245 (34%)	132 (54%)	119,60 (25%)
UY-01	13,52 (17%)	24 046 (73%)	500 (69%)	300 (60%)	73,34 (16%)
US-03	10,24 (13%)	26 346 (80%)	483 (67%)	296 (61%)	101,59 (22%)

botnets, que têm baixo volume de mensagens por IP de origem. Os poucos endereços IP que utilizam HTTP/SOCKS (1,50% do total) têm volume muito maior de mensagens, típico de servidores de *spam* dedicados.

A tabela 2 mostra a quantidade de mensagens, o número de IPs de origem distintos, o número de ASes de origem e o volume das mensagens recebidas observados em cada *honeypot*. Note que existe sobreposição de IPs entre *honeypots*, o que indica que *spammers* conectam-se a mais de um *honeypot*.

2.2. Inferência das rotas trafegadas pelo *spam*

Para inferir as rotas percorridas pelas mensagens de *spam*, primeiro coletamos medições utilizando *traceroute* e depois mapeamos os endereços IP nos traceroutes em sistemas autônomos (ASes).

Usamos as plataformas RIPE Atlas e PlanetLab para medir as rotas que seriam trafegadas pelas mensagens de *spam* dos *honeypots* até seus destinos, caso as mensagens tivessem sido entregues. Estas plataformas possuem monitores em milhares de redes ao redor do globo, inclusive nas redes dos *honeypots*. Como os ASes onde os *honeypots* estão hospedados são ASes locais e de ponta, os dispositivos RIPE Atlas e PlanetLab provavelmente usam a mesma rota dos *honeypots* até os *spammers* [Mühlbauer et al. 2007]. Durante nossas medições observamos que 29% dos domínios de destino não puderam ser resolvidos ou não possuem servidor de e-mail configurado no DNS (i.e., o domínio não configura registro MX no DNS), o que impossibilita realizar uma medição *traceroute* para o servidor de e-mail. Isto pode acontecer para destinatários gerados (semi-)automaticamente. Como um *relay* não encaminharia mensagens de e-mail nesse caso, ignoramos estes domínios.

Para medir as rotas trafegadas pelas mensagens de *spam* dos *spammers* até os *honeypots*, seria necessário ter acesso a dispositivos próximos aos *spammers*. Essa abor-

dagem não é prática pois as mensagens de *spam* são enviadas de 722 ASes distintos, 58,12% deles não cobertos por nós PlanetLab ou RIPE Atlas. O número de ASes não cobertos, por *honeypot*, pode ser visto na tabela 2. Para contornar este problema realizamos medições dos *honeypots* até os *spammers*, utilizando as plataformas RIPE Atlas e PlanetLab, e assumimos que a rota dos *spammers* até os *honeypots* é simétrica. Uma limitação desta abordagem é que rotas na Internet são frequentemente assimétricas [He et al. 2005].

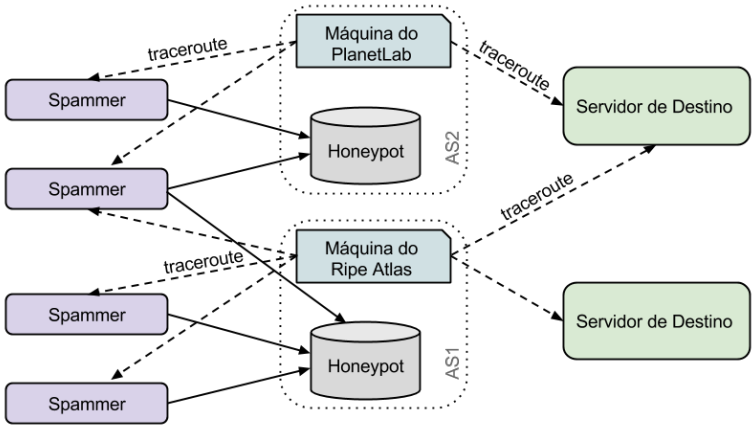


Figura 1. Arquitetura utilizada na coleta das medições de *traceroute*

A figura 1 ilustra a plataforma que implantamos para coletar medições de *traceroute*. A figura ilustra que usamos coletores PlanetLab e RIPE Atlas distintos dentro do AS de cada *honeypot* bem como a sobreposição de *spammers* e destinatários entre os *honeypots*. Por último, a direção das arestas pontilhadas mostra o sentido das medições *traceroute* e as arestas sólidas mostram o sentido das mensagens de *spam*.

Como nosso estudo é baseado nas relações entre ASes, precisamos mapear endereços IP obtidos nas medições com *traceroute* em seus respectivos ASes. Utilizamos as bases de mapeamento de endereço IP em número de AS do iPlane [Madhyastha et al. 2006] e do Team Cymru.[3] Após aplicar as bases de dados para converter endereços IP em ASes, nós substituímos sequências do mesmo AS por uma única ocorrência e aplicamos heurísticas propostas por [Chen et al. 2009] para tratar erros de mapeamento causados por endereços IP que não estão nas bases de dados. Em particular, se um endereço IP x não foi mapeado mas está cercado por endereços IP mapeados para um mesmo AS_1, e.g., $[\cdots, \mathrm{AS}_1, x, \mathrm{AS}_1, \cdots]$, então substituímos x por AS_1. A fração de rotas em nosso conjunto de dados com erros de mapeamento entre sistemas autônomos distintos, e.g., $[\cdots, \mathrm{AS}_1, x, \mathrm{AS}_2, \cdots]$, varia de 2 a 25% dependendo do *honeypot*. Nestes casos simplesmente ignoramos o erro de mapeamento e consideramos que os dados trafegam de AS_1 para AS_2. Esta heurística impacta a completude dos nossos resultados (quando ignoramos um AS), mas *não* impacta a correção dos resultados (pois os dados trafegaram entre os ASes considerados, mesmo que indiretamente). Obtemos a tabela de roteamento BGP do AS que hospeda um dos *honeypots* e verificamos que 89% das rotas convertidas de endereços IP para ASes são idênticas às rotas BGP (98% tem diferença de até um AS).

[3]http://www.team-cymru.org/Services/ip-to-asn.html

Utilizamos a base de dados de relações (políticas de roteamento) entre ASes disponibilizada pela CAIDA [Luckie et al. 2013]. Neste artigo assumimos que ASes clientes pagam a ASes provedores para obter conectividade com o resto da Internet e que ASes parceiros (peer) trocam tráfego sem custo para ambas as partes. Estas suposições são alinhadas com suposições em outros trabalhos na literatura [Gao 2001, Luckie et al. 2013, Oliveira et al. 2010].

3. Avaliação do custo do tráfego de *spam*

Nesta seção, apresentamos os resultados obtidos através da metodologia apresentada na seção 2. Para não comprometer a identidade dos *honeypots* omitimos suas localizações exatas e anonimizamos as redes hospedeiras.[4] Primeiramente apresentamos um estudo de caso de um *honeypot* no Brasil, o BR-02 (seção 3.1). Este estudo de caso, além de ilustrar como a análise de resultados foi feita, revela importantes conclusões práticas sobre como filtros bem colocados na rede podem bloquear grande parte do *spam* mais próximo à origem. Por fim generalizamos os resultados encontrados para BR-02 para os outros cinco *honeypots*, mostrando que os resultados encontrados são aplicáveis ao tráfego *spam* de forma geral e identificando quais são as redes mais oneradas pelo tráfego de *spam* (seção 3.2).

3.1. Estudo de caso do *honeypot* BR-02

As rotas dos mapas das figuras 2 e 3 sintetizam os ASes mais utilizados nas rotas obtidas pelos traceroutes coletados através de uma máquina do PlanetLab no AS do *honeypot* BR-02. O mapa da figura 2 mostra os ASes que mais apareceram nas rotas entre a máquina do PlanetLab e as máquinas que enviaram *spam*. A direção das arestas mostra o sentido do *traceroute* e o tipo da linha a relação comercial. As variáveis sobre as arestas mostram a porcentagem do volume de mensagens de *spam* indo em direção ao *honeypot* que passa pela aresta.

O mapa da figura 2 contém alguns pontos interessantes. O primeiro é a aresta entre o AS do *honeypot* BR-02 e um de seus provedores, a Level 3 (AS3356), onde cerca de 77% do volume do tráfego, é trafegado. Além disso, existe um grande número de endereços IP de origem (24107) cujas mensagens chegam ao *honeypot* pela Level 3. Neste caso, conforme será detalhado na seção 3.2, é o enlace que mais gera custos para o AS do *honeypot* BR-02. O segundo ponto de interesse é a aresta entre o *honeypot* BR-02 e a Hurricane Electric (AS6939), onde 20% do volume total é trafegado. Neste enlace, tem-se um baixo número de endereços IP de origem que enviam um grande número de mensagens (28,4% do total). Realizando uma análise mais profunda, verificamos que 75 dos 82 endereços IP utilizam o protocolo SOCKS, indicando que esse enlace é muito usado por tráfego de *spam* gerado por servidores dedicados.

Finalmente, o terceiro ponto de interesse é entre a HiNetUSA (de Taiwan, AS9680) e a HiNet (AS3462), que enviam cerca de 51% do volume trafegado. Essa rota é interessante visto que a HiNetUSA recebe o *spam* da HiNet e pode estar dividindo esse tráfego em várias rotas, de forma a baratear despesas com tráfego de rede. Isso acontece pelo fato que, em grande parte dos contratos entre ASes, o custo pela

[4]Como nosso objetivo é continuar monitorando a rede de forma a estudar e detectar mudanças no comportamento dos *spammers*, precisamos proteger a identidade dos *honeypots* para preservar sua utilidade.

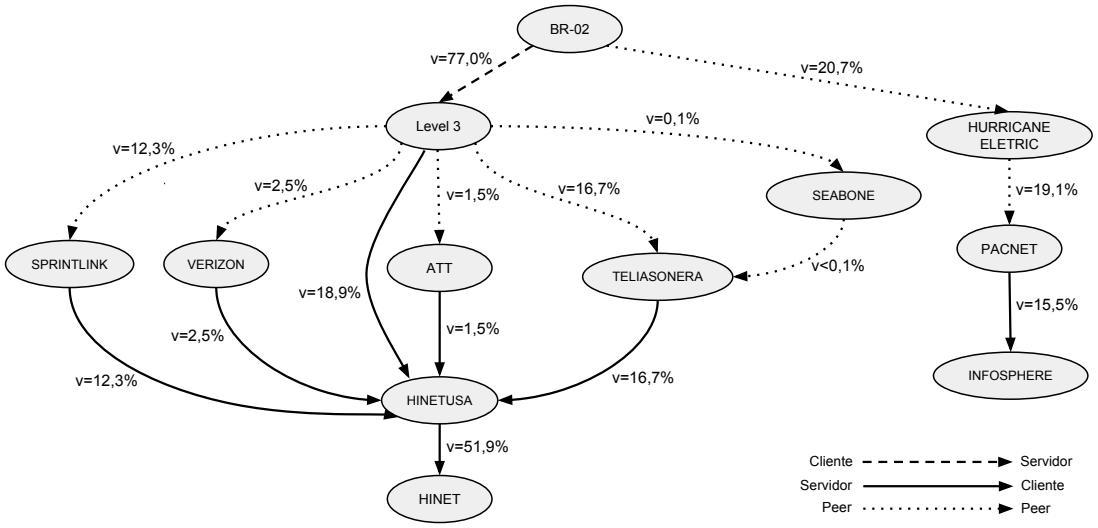


Figura 2. Mapa dos ASes mais utilizados entre o *honeypot* e as máquinas que enviam *spam*.

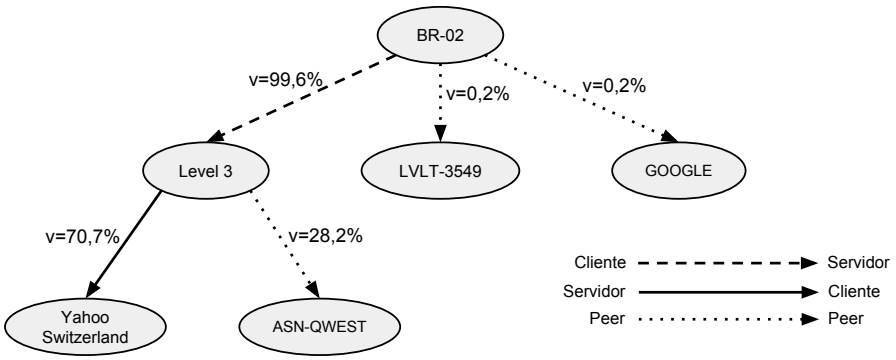


Figura 3. Mapa dos ASes mais utilizados entre o *honeypot* e os domínios de destino.

banda é proporcional ao 95º percentil da banda consumida em intervalos de cinco minutos [Dimitropoulos et al. 2009, Valancius et al. 2011]. Em uma análise minuciosa, verificamos que os prefixos anunciados pela HiNetUSA para seus provedores[5] são disjuntos, o que distribui e reduz o 95º percentil do tráfego. Entretanto, não pode-se afirmar que a HiNetUSA está hospedando *spammers* visto que aproximadamente 98% dos endereços IP que utilizam essa rota usam o protocolo SMTP que, conforme explicado na seção 2.1, geralmente fazem parte de *botnets*.

O mapa da figura 3 é similar ao mapa da figura 2, mas mostra o mapa dos ASes que carregariam o maior volume de tráfego de *spam* enviado pelo *honeypot* até os destinatários, caso as mensagens tivessem sido encaminhadas. Conforme será analisado na próxima seção, o volume de saída do *honeypot* é maior que o de entrada, pois mensagens com vários destinos precisam ser replicadas pelo menos uma vez para cada domínio com destinatários.[6] Em outras palavras, o volume de tráfego que circula entre os ASes mostrados na figura 3 é maior que o tráfego que circula entre os ASes mostrados na fi-

[5]Sprint (AS1239), Verizon (AS701), Telia (AS1299), AT&T (AS7018) e Level 3 (AS3356).

Tabela 3. ASes que mais pagam/recebem no BR-02 do *spammer* ao *honeypot*

MAIS PAGAM (ASN)	VOLUME (GB)	MAIS RECEBEM (ASN)	VOLUME (GB)
AS *Honeypot* BR-02, BR	66,5	Level 3, US (3356)	97,0
HiNetUSA, TW (9680)	44,1	HiNetUSA, TW (9680)	44,1
HiNet, TW (3462)	44,1	PacNet, HK (10026)	16,2
InfoSphere NTT PC, JP (2514)	14,0	TeliaSonera, SE (1299)	14,2
Internet INT Japan, JP (2497)	11,9	SprintLink (1239)	10,4

gura 2. Para calcular o volume de mensagens do *honeypot* aos destinatários do *spam* supomos que apenas uma cópia da mensagem seria enviada para todos os recipientes em um mesmo servidor de destino, como recomendado no RFC 821. O tráfego poderia ser ainda maior caso a recomendação do RFC não fosse seguida.

De acordo com o mapa da figura 3, temos que o *honeypot* BR-02 transmitiria 99% do volume de *spam* de volta pela Level 3. Esse volume, 28 vezes maior que o volume que chega ao *honeypot*, aumenta ainda mais os custos de comunicação. Além disso, grande parte deste volume (70%) é repassado para o Yahoo, um dos maiores operadores de e-mail, possivelmente gerando grandes custos para esta rede. Portanto, um possível uso de filtros de *spam* no AS que hospeda o *honeypot* BR-02 evitaria replicação de mensagens e reduziria custos de rede não só no seu próprio AS mas também para o Yahoo. Note que após o *honeypot* encaminhar as mensagens de *spam*, o Yahoo não pode filtrá-las antes de recebê-las.

3.1.1. Análise de custos do BR-02

As rotas dos mapas 2 e 3 incitam um questionamento pertinente sobre quem paga e quem recebe pelo tráfego de *spam* gerado. As tabelas 3 e 4 mostram os principais ASes que são onerados e os principais ASes que lucram em relação ao volume enviado. Para realizar essa análise, consideramos as relações comerciais entre os ASes dessas rotas. Para relações cliente-provedor, contabilizamos custos para o cliente e lucro para o provedor proporcionais à quantidade de tráfego. Note que um provedor pode receber por volume de tráfego maior do que o volume recebido pelo *honeypot* caso ele receba o tráfego de um cliente e repasse a outro cliente. Para relações de parceria e relações não inferidas, nenhum tipo de contagem é realizado. Neste caso, o *honeypot* pode pagar por menos tráfego do que recebido se parte do tráfego chegar por um AS parceiro. Medimos o volume de tráfego por que desconhecemos a relação contratual entre os sistemas autônomos (i.e., precificação da banda e condições de uso); não podemos estimar o valor monetário que um AS está pagando ao outro. Notamos que apesar desta ser uma limitação do nosso trabalho, esta informação é sensível e provavelmente protegida por acordo de sigilo (NDAs) entre as partes.

Realizando uma análise do tráfego que sai do *spammer* em direção ao *honeypot* BR-02, a tabela 3 mostra os cinco ASes que mais pagaram e os que mais lucraram com este tráfego. Como esperado, o AS que mais perdeu, em tráfego, foi o AS do *honeypot*

[6]Por exemplo, um e-mail destinado a elverton@dcc.ufmg.br, dorgival@dcc.ufmg.br e jessen@cert.br precisa ser enviado aos servidores de e-mail do DCC/UFMG e do CERT.br.

Tabela 4. ASes que mais pagam/recebem no BR-02 do *honeypot* ao servidor de destino

MAIS PAGAM (ASN)	VOLUME (GB)	MAIS RECEBEM (ASN)	VOLUME (GB)
AS *Honeypot* BR-02, BR	1.898,7	Level 3, US (3356)	3.247,3
Yahoo-Switzerland, CH (42173)	1.346,4	Qwest, US (209)	538,1
KDDi Corp, JP (2516)	538,4	NTT-Comm, US (2914)	4,0
OCN NTT, JP (4713)	4,2	LVLT-3549, US (3549)	2,8
Microsoft Corp, US (8075)	2,7	TiNet-Backbone, DE (3257)	1,4

BR-02 seguido pela HiNetUSA. Entre os ASes que mais receberam, temos a Level 3 como maior beneficiária seguido da HiNetUSA. Neste caso, a HiNetUSA, que divide o tráfego entre seus provedores conforme analisado na seção anterior, está entre os que mais ganharam recebendo 44,1 GB de seus clientes e os que mais perderam enviando a mesma quantidade para seus provedores, ficando com saldo nulo. Entretanto, na prática, este resultado indica que a HiNetUSA obteve lucro, visto que o valor que um AS cobra para seus clientes é maior que o valor que ele paga para seus provedores. Dessa forma, o balanceamento de rotas realizado neste AS pode estar sendo feito para tentar tornar os custos menores.

Fazendo um estudo sobre o volume do tráfego que sairia do *honeypot* BR-02 em direção aos servidores de email, a tabela 4 mostra os ASes que mais perderam e os que mais ganharam com o tráfego. Conforme mencionado na seção anterior, as mensagens que chegam ao servidor SMTP têm uma cópia enviada para cada domínio distinto nos destinatários das mensagens. Isso representa uma amplificação no tráfego, conforme pode ser visto comparando o volume de tráfego entre as tabelas 3 e 4, onde o AS do *honeypot* BR-02 que antes pagava por um volume de 66,5 GB, passa a pagar por mais de 1.8 TB, representando um aumento de, aproximadamente, 2800%. Além disso, redes com servidores de destino como o Yahoo (AS42173), passam a ser oneradas por um volume significativo de tráfego. Essa amplificação afeta também os que mais recebem, como é o caso da Level 3, com um saldo positivo para mais de 3 TB recebidos. A Qwest (AS209) é outro AS que também lucraria caso as mensagens de *spam* do *honeypot* BR-02 fossem entregues aos destinos.

Este estudo de caso mostra a importância do esforço no combate ao *spam* na origem. Em particular, grandes provedores como Level 3, Qwest e Telia não têm incentivos para implantar filtros contra tráfego de *spam*. Mesmo sem cooperação de grandes provedores, redes de borda podem cooperar para filtrar o tráfego de *spam*. Em particular, se filtros fossem posicionados na HiNet, por exemplo, reduziriam o tráfego que chega ao *honeypot* em quase 52%. De forma similar, outro filtro na InfoSphere (AS2514) reduziria o tráfego de *spam* em mais 15%. Como consequência, esse volume de *spam* não chegaria a servidores de e-mail (como o *honeypot*) e não seria replicado para reenvio aos servidores de destino, reduzindo drasticamente o tráfego enviado aos destinatários e evitando desperdício de recursos. Por último, notamos que mesmo sem cooperação das redes que enviam grande quantidade de *spam* (como HiNet e InfoSphere), ainda seria possível filtrar aproximadamente 37% do tráfego em direção ao *honeypot* e parte do tráfego replicado para reenviar as mensagens até os destinatários.

Tabela 5. ASes que mais pagam e recebem, por *honeypot*, pelo tráfego do *spammer* ao *honeypot*

ID	MAIS PAGAM (ASN)	VOLUME (GB)	MAIS RECEBEM (ASN)	VOLUME (GB)
	AS *Honeypot* BR-01	90,5	Embratel,BR (4230)	90,6
	Embratel, BR (4230)	48,0	Verizon, US (701)	39,7
BR-01	HiNetUSA, TW (9680)	24,0	HiNetUSA, TW (9680)	24,0
	HiNet, TW (3462)	24,0	SprintLink, US (1239)	17,5
	ROSTelecom, RU (12389)	10,2	TiNet-Backbone, DE (3257)	11,9
	AS *Honeypot* NL-01	86,7	TiNet-Backbone, DE (3257)	69,9
	HiNet, TW (3462)	30,4	HiNetUSA, TW (9680)	30,4
NL-01	HiNetUSA, TW (9680)	30,3	KPN Eurorings, NL (286)	20,7
	InfoSphere, JP (2514)	22,8	PacNet, HK (10026)	20,4
	OCN NTT, JP (4713)	5,6	TeliaNet, SE (1299)	16,3
	AS *Honeypot* US-03	58,4	XO Comm, US (2828)	32,1
	HiNet, TW (3462)	22,5	TWCable-Backbone, US (7843)	26,3
US-03	HiNetUSA, TW (9680)	19,7	Level 3, US (3356)	25,7
	CW CABLE AND WIRELESS WW, GB (1273)	5,4	HiNetUSA, TW (9680)	22,5
	TWCable-Backbone, US (7843)	5,3	TeliaNet, SE (1299)	13,7
	AS *Honeypot* UY-01	77,2	SprintLink, US (1239)	111,2
	HiNetUSA, TW (9680)	23,0	HiNetUSA, TW (9680)	23,0
UY-01	HiNet, TW (3462)	23,0	NTT-Comm, US (2914)	15,0
	InfoSphere, JP (2514)	12,3	TWGATE-AP, TW (9505)	9,4
	TWGATE-AP, TW (9505)	9,4	ROSTelecom, RU (12389)	9,0

3.2. Análise geral dos *honeypots*

Nesta seção, estendemos o estudo de caso feito na seção anterior para os outros quatro *honeypots* (BR-01, NL-01, US-03 e UY-01). A tabela 5 mostra os ASes que mais foram onerados e os que mais receberam, por *honeypot*, nas rotas trafegadas pelas mensagens enviadas por *spammers* aos *honeypots*. Como pode-se notar, em todos os casos, os ASes dos *honeypots* são os que mais são onerados indicando que, na prática, ASes que recebem *spam* são os mais prejudicados. De forma similar, também vemos que os ASes que enviam a maior quantidade de *spam*, como a HiNet, também é onerada.

Um outro fato interessante é a presença da HiNetUSA em todos os quatro *honeypots* da tabela 5, dentre os mais onerados e também dentre os que mais recebem. Como observado para o *honeypot* BR-02, grande parte dos *spammers* que utilizam rotas através da HiNetUSA usam o protocolo SMTP, em todos os *honeypots* (não mostrado). Isto reforça o fato que o *spam* enviado através da HiNetUSA é oriundo de *botnets* e não podemos afirmar que este AS está sendo conivente. Porém podemos afirmar que filtros de *spam* instalados na HiNet reduziriam o tráfego de *spam* de forma global.

A tabela 6 mostra o mesmo dado que a tabela 5 para as rotas trafegadas pelas mensagens enviadas do *honeypot* ao servidor de destino. A relação entre o tráfego pago pelo *honeypot* na tabela 5, e o tráfego pago pelo *honeypot* na 6 segue um padrão bastante pertinente. Como pode-se observar, o volume pago pelos *honeypots* é muito menor na tabela 5 do que na tabela 6. Isso se deve ao fato discutido sobre a amplificação do tráfego que as mensagens de *spam* podem ocasionar ao atingir um servidor SMTP. Neste caso, o volume pago pelo *honeypot* na saída pode aumentar mais de 3000%, como no caso do *honeypot* NL-01. Em posse dessa visão mais global, é possível mostrar como o uso de filtros de *spam* que impeçam mensagens de atingir um servidor SMTP podem reduzir o

Tabela 6. ASes que mais pagam e recebem, por *honeypot*, por tráfego do *honeypot* aos servidores de destino

ID	MAIS PAGAM (ASN)	VOLUME (GB)	MAIS RECEBEM (ASN)	VOLUME (GB)
	AS *Honeypot* BR-01	2.230,9	Embratel, BR (4230)	2.230,9
	MailRu, RU (47764)	475,0	RETN Limited, UA (9002)	475,0
BR-01	Embratel, BR (4230)	61,4	Verizon, US (701)	104,9
	HiNetUSA, TW (9680)	47,3	HiNetUSA, US (9680)	47,3
	HiNet, TW (3462)	47,3	BTN, US (3491)	8,8
	AS *Honeypot* NL-01	3.866,8	POPPE, US (4525)	177,8
	Yahoo-JP-AS-AP, JP (24572)	177,8	GigaInfra, JP (17676)	177,8
NL-01	GigaInfra, JP (17676)	177,8	ROSTelecom, RU (12389)	156,2
	RTComm, RU (8342)	166,1	Verizon, US (702)	14,5
	Sovam, RU (3216)	14,5	IpTransit, US (46786)	12,6
	AS *Honeypot* US-03	146,0	XO-AS15, US (2828)	136,1
	TransTelecom, RU (20485)	56,4	Level 3, US (3356)	125,6
US-03	HiNet, TW (3462)	49,5	HiNetUSA, TW (9680)	49,5
	HiNetUSA, TW (9680)	49,5	Verizon, US (701)	48,9
	MailRu, RU (47764)	47,5	TransTelecom, RU (20485)	47,5
	AS *Honeypot* UY-01	1.478,2	SprintLink, US (1239)	1.486,5
	SeedNet, TW (4780)	9,0	Level 3, US (3356)	9,3
UY-01	BTN-ASN, US (3491)	8,5	BTN-ASN, US (3491)	8,5
	HiNet, TW (3462)	3,4	HiNetUSA, TW (9680)	3,3
	HiNetUSA, TW (9680)	3,3	ChinaNet, CN (4134)	2,6

tráfego da rede. O impacto é bem maior que os filtros utilizados nos servidores de destino que não impedem todo este gasto de recursos.

Por último, apontamos alguns ASes que apresentam um comportamento interessante, pois, apesar de estarem entre os cinco ASes que mais pagam pelo tráfego de *spam*, a quantidade que eles recebem é sempre maior. A Embratel é um ótimo estudo de caso neste cenário, visto que ela apresenta este comportamento tanto nas rotas entre o *spammer* e o *honeypot* quanto nas rotas entre o *honeypot* e os servidores de destino. Por exemplo, no *honeypot* BR-01, ela aparece na tabela 5 pagando por 48,0 GB de tráfego, mas recebe por 90,6 GB. De forma similar, a Embratel paga por 61,4 GB de tráfego enviado do *honeypot* aos destinatários (tabela 6), mas recebe por 2.230,9 GB de tráfego. A Embratel consegue reduzir custos e obter lucro trafegando mensagens de *spam* graças a uma parceria de troca de tráfego com a Hurricane Electric. A TW Cable apresenta um comportamento semelhante, mas em uma proporção menor, recebendo quatro vezes mais do que paga, como pode ser observado na tabela 5.

4. Trabalhos Relacionados

Medição de Rotas e Mapeamento em Sistemas Autônomos. Neste trabalho fazemos uso de técnicas de mapeamento topológico para obter informações sobre as rotas por onde as mensagens de spam foram transmitidas. Bases de dados para mapeamento de endereços IP em ASes [Madhyastha et al. 2006] são construídas mapeando prefixos IP para o AS que originou o anúncio BGP daquele prefixo. Para maximizar a cobertura do espaço de endereçamento IP, estas bases utilizam anúncios de rotas coletados por diversos roteadores BGP conectados à Internet (como os roteadores dos projetos RouteViews e RIPE RIS) bem como informações de alocação de prefixos de registros como ARIN, RIPE e LAC-

NIC. [Chen et al. 2009] propuseram diversas heurísticas para correção de erros comuns inseridos pelo mapeamento de endereços IPs em ASes. Em particular, eles observaram que endereços IPs pertencentes a pontos de troca de tráfego (PTT) podem aparecer entre dois ASes membros do PTT e que ASes podem usar endereços IPs "emprestados" de outros ASes; ambos casos levam à criação de parcerias falsas. Em nosso trabalho implementamos e utilizamos apenas algumas das heurísticas propostas por [Chen et al. 2009].

Os primeiros algoritmos para identificação de relações entre ASes foram propostos por [Gao 2001]; neste trabalho utilizamos a base de dados mais recente à qual temos acesso, disponibilizada pela CAIDA [Luckie et al. 2013]. O algoritmo de inferência da CAIDA utiliza várias regras que capturam técnicas de engenharia de tráfego, práticas de mercado e políticas de roteamento. A base de dados da CAIDA tem precisão de 95% [Luckie et al. 2013] e continua sendo aprimorada [Giotsas et al. 2014].

Caracterização de Tráfego de Spam na Internet. Existem vários trabalhos sobre caracterização de tráfego na Internet.[7] Mais relacionados a este artigo são trabalhos sobre a caracterização de propriedades do tráfego de *spam*. Por exemplo, [Ramachandran and Feamster 2006] mostram que diversos elementos do tráfego de spam podem ser usados para tentar identificar sua transmissão através da rede. Mais importante do ponto de vista deste estudo são trabalhos que usam dados de rotas para identificar ASes maliciosos [Konte and Feamster 2011, Ramachandran and Feamster 2006]. De forma similar, neste trabalho identificamos práticas de engenharia de tráfego suspeitas.

Custos de Spam e Tráfego na Internet. Sobre a questão de custo do spam, [Anderson et al. 2007] apresentam uma análise do custo da infra-estrutura de envio de spam, bem como do custo de conectividade para o spammer, mas não há uma análise do impacto desse tráfego no custo operacional dos demais ASes, como buscamos fazer neste artigo. Também relacionados a esse tema são trabalhos que discutem propriedades do modelo de cobrança comumente utilizado para trânsito na Internet, baseado no 95º percentil do tráfego [Stanojevic et al. 2010, Dimitropoulos et al. 2009]. Estes estudos mostram que cobrança pelo 95º percentil é sensível à parametrização (e.g., duração do intervalo de agregação de tráfego) e que a sensibilidade é maior para redes que trafegam poucos dados (e.g., redes de borda). Mesmo trabalhos mais recentes que consideram novos modelos alternativos para cobrança por trânsito na Internet consideram o volume de tráfego e os enlaces por onde o dado trafega [Valancius et al. 2011]. Neste trabalho mostramos que redes de borda são as mais oneradas pelo tráfego de *spam* e discutimos como ASes aparentam tentar minimizar o custo para enviar o tráfego de *spam* e como redes de borda são prejudicadas.

Também relacionados às nossas análises são propostas para filtrar mensagens de e-mail na origem e reduzir o custo de propagação de *spam* pela Internet [Las-Casas et al. 2013]. Nossa metodologia permite identificar quais redes têm incentivos para implantar tais filtros, facilitando a cooperação e implantação destas técnicas.

5. Conclusão e Trabalhos Futuros

Neste trabalho, objetivamos lançar mais luz sobre os custos gerados pela prática de *spamming* através de mensagens capturadas por cinco *honeypots* em redes distintas. Realiza-

[7]Por exemplo, o relatório *Global Internet Phenomena* da Sandvine apresenta e discute tendências de tráfego na Internet a cada seis meses (https://www.sandvine.com/trends/global-internet-phenomena/).

mos medições com *traceroutes* a partir das plataformas RIPE Atlas e PlanetLab que, combinadas com técnicas para mapeamento de endereços IP em sistemas autônomos (AS), permitiram inferir as rotas utilizadas pelos *spammers*. Além disso, utilizamos resultados recentes sobre inferências de relações comerciais entre ASes para quantificar os custos gerados pelo tráfego de *spam* em cada rede.

Nossos resultados mostram que algumas redes são sistematicamente oneradas pelo tráfego de *spam* mas que podem cooperar para reduzir bastante o tráfego filtrando mensagens de *spam* na origem. Além disso, mostramos que ASes que enviam grande quantidade de *spam* aparentam utilizar práticas de engenharia de tráfego para reduzir custo. Por fim, quantificamos o efeito amplificador que as mensagens de *spam* têm ao alcançar servidores SMTP e a redução drástica deste tráfego de *spam* caso fossem instalados filtros para impedir que essas mensagens atinjam tais servidores.

Como trabalho futuro pretendemos realizar essas análises para diferentes períodos, de forma a entender a dinâmica dos custos do tráfego de *spam*, por exemplo, em função de mudanças de roteamento. Além disso, com monitoramento constante podemos identificar quais ASes estão tendo os seus custos aumentados e alertá-los, servindo como incentivo para que os mesmos utilizem filtros para reduzir o volume de *spam* trafegado.

Agradecimentos

Este trabalho foi parcialmente financiado por NIC.BR, Fapemig, CAPES, CNPq e InWeb. Gostaríamos de agradecer, também, ao Matt Calder, da Universidade do Sul da Califórnia (EUA), pelo scripts fornecidos para a utilização da plataforma RIPE Atlas.

Referências

Anderson, D. S., Fleizach, C., Savage, S., and Voelker, G. M. (2007). Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *USENIX Security Symposium*.

Chen, K., Choffnes, D. R., Potharaju, R., Chen, Y., Bustamante, F. E., Pei, D., and Zhao, Y. (2009). Where the Sidewalk Ends: Extending the Internet AS Graph Using Traceroutes from P2P Users. In *Proc. ACM CoNEXT*.

Dimitropoulos, X., Hurley, P., Kind, A., and Stoecklin, M. (2009). On the 95-Percentile Billing Method. In *Proc. PAM*.

Gao, L. (2001). On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Trans. Netw.*, 9(6):733–745.

Giotsas, V., Luckie, M., Huffaker, B., and kc claffy (2014). Inferring Complex AS Relationships. In *Proc. IMC*.

He, Y., Faloutsos, M., Krishnamurthy, S., and Huffaker, B. (2005). On Routing Asymmetry in the Internet. In *Proc. IEEE GLOBECOM*.

Konte, M. and Feamster, N. (2011). Wide-area Routing Dynamics of Malicious Networks. In *Proc. ACM SIGCOMM*.

Las-Casas, P. H. B., Guedes, D., Almeida, J. M., Ziviani, A., and Marques-Neto, H. T. (2013). SpaDeS: Detecting Spammers at the Source Network. *Computer Networks*, 57(2):526–539.

Luckie, M., Huffaker, B., Claffy, K., Dhamdhere, A., and Giotsas, V. (2013). AS Relationships, Customer Cones, and Validation. In *Proc. IMC*.

Madhyastha, H., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., and Venkataramani, A. (2006). iPlane: an Information Plane for Distributed Services. In *Proc. USENIX OSDI*.

Mühlbauer, W., Uhlig, S., Fu, B., Meulle, M., and Maennel, O. (2007). In Search for an Appropriate Granularity to Model Routing Policies. In *Proc. ACM SIGCOMM*.

Newman, M. E. J., Forrest, S., and Balthrop, J. (2002). Email Networks and the Spread of Computer Viruses. *Phys. Rev. E*, 66(3):035101.

Oliveira, R., Pei, D., Willinger, W., Zhang, B., and Zhang, L. (2010). Quantifying the Completeness of the Observed Internet AS-level Structure. *IEEE/ACM Trans. Netw.*, 18(1):109–122.

Orman, H. (2013). The Compleat Story of Phish. *IEEE Internet Computing*, 17(1):87–91.

Ramachandran, A. and Feamster, N. (2006). Understanding the Network-level Behavior of Spammers. In *Proc. ACM SIGCOMM*.

Rao, J. M. and Reiley, D. H. (2012). The Economics of Spam. *The Journal of Economic Perspectives*, 26(3):87–110.

Sipior, J. C., Ward, B. T., and Bonner, P. G. (2004). Should Spam Be on the Menu? *Communications of the ACM*, 47(6):59–63.

Stanojevic, R., Laoutaris, N., and Rodriguez, P. (2010). On Economic Heavy Hitters: Shapley Value Analysis of 95th-percentile Pricing. In *Proc. IMC*.

Symantec (2014). Internet Security Threat Report, Volume 19. Online.

Valancius, V., Lumezanu, C., Feamster, N., Johari, R., and Vazirani, V. V. (2011). How Many Tiers? Pricing in the Internet Transit Market. In *Proc. ACM SIGCOMM*.

Equilibrando Energia, Redundância e Desempenho em Redes de Centros de Dados

Antônio Cleber de S. Araújo[1], **Leobino N. Sampaio**[1], **Artur Ziviani**[2]

[1]Mestrado Multinstucional em Ciência da Computação (MMCC)
Instituto de Matemática – Universidade Federal da Bahia (UFBA)
Salvador, BA - Brasil

[2]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis, RJ - Brasil

{antonio.cleber,leobino}@ufba.br, ziviani@lncc.br

Abstract. *The current large-scale datacenters typically adopt redundancy of servers and communication devices for increased reliability and availability. Highly redundant infrastructure, however, is one of the challenges of the area due to high energy consumption. This paper presents the Beep, an energy-efficient strategy for datacenter networks based on the Fat Tree topology. Our strategy uses multipaths and the global overview offered by software-defined networks to balance the energy efficiency, the equipment redundancy level, and the performance gain when dealing with the traffic demands. The BEEP was implemented in an OpenFlow network and the multipaths using MPTCP (Multipath TCP). The experimental results in variants of the Fat Tree topology show gains in the energy efficiency with the strategy in the order of 30% to 45%, additionally to a better utilization of the available bandwidth, as more alternative paths are available.*

Resumo. *Os grandes centros de dados atuais tipicamente adotam redundância de servidores e equipamentos de comunicação para aumento de sua confiabilidade e disponibilidade. Infraestrutura altamente redundante, contudo, consiste num dos desafios da área devido ao alto consumo de energia. Este artigo apresenta a BEEP, uma estratégia energeticamente eficiente para redes de centro de dados baseadas na topologia Fat Tree. Nossa estratégia faz uso de múltiplos caminhos e da visão global oferecida por redes definidas por software para equilibrar eficiência energética, nível de redundância dos equipamentos e ganho de desempenho no atendimento às demandas de tráfego. A BEEP foi implementada através de uma rede OpenFlow e os múltiplos caminhos por MPTCP (Multipath TCP). Resultados experimentais em variantes da topologia Fat Tree demonstraram ganhos de eficiência energética com a estratégia na ordem de 30% a 45%, além de melhoria na utilização da largura de banda disponível conforme haja mais caminhos alternativos disponíveis.*

1. Introdução

Em anos recentes, os elevados requisitos de confiabilidade e disponibilidade dos serviços providos pelos atuais centros de dados de larga escala têm motivado a comunidade de pesquisa a propor novas soluções no intuito de melhorar seus desempenhos. Em geral, as

propostas envolvem a adoção de topologias de redes projetadas para garantir um mínimo de redundância na infraestrutura de servidores e equipamentos [Greenberg et al., 2009, Guo et al., 2009].

A redundância dos equipamentos pode, contudo, acarretar aumentos significativos no consumo de energia. Este aumento é ratificado, principalmente, pelo aumento da utilização de *switches* com velocidade de 10 Gbps em centros de dados [Bilal et al., 2014]. Estudos estimam que a infraestrutura de rede dos EUA usa 24 TWh por ano [Nordman, 2007], custando US$ 24 bilhões. Isso inclui *switches*, roteadores, pontos de acesso e placas de rede em *hosts* e servidores. Por tais motivos, estão em curso esforços para reduzir o consumo de energia de todos os componentes da Internet como os padrões *EnergyStar* e uma força-tarefa do IEEE para prover eficiência energética em redes *Ethernet* [IEEE, 2014].

As soluções de economia de energia passam muitas vezes pela adoção de estratégias de gerenciamento autonômico que buscam desligar os equipamentos com base na demanda de tráfego [Hammadi e Mhamdi, 2014]. O problema, entretanto, é que, muitas vezes, tais estratégias comprometem a redundância mínima necessária, assim como, o desempenho das suas redes em prol de ganhos no consumo energético.

Motivado por estas questões, este trabalho apresenta uma estratégia de gerenciamento de energia para redes de centro de dados baseadas em topologias do tipo *Fat Tree* - BEEP (do inglês, *Balanced Energy, rEdundancy, and Performance*). A estratégia busca equilibrar o consumo de energia, a redundância dos equipamentos e o desempenho da rede através de dois algoritmos que fazem uso da visão global da rede e exploram os múltiplos caminhos de comunicação entre os *hosts*. Nossa solução foi implementada através de uma rede *OpenFlow* [McKeown et al., 2008] e os múltiplos caminhos por MPTCP (*Multipath* TCP) [Barré et al., 2011]. Resultados experimentais em variantes da topologia *Fat Tree* demonstraram ganhos de eficiência energética com a estratégia BEEP na ordem de 30% a 45%, além de melhoria na utilização da largura de banda disponível conforme haja mais caminhos alternativos disponíveis.

O presente artigo está organizado da seguinte forma. Na Seção 2 são discutidos trabalhos relacionados. A Seção 3 apresenta a solução proposta. Na Seção 4 são descritos os experimentos realizados com a plataforma *OpenFlow* e MPTCP, assim como, são analisados os resultados experimentais obtidos. Por fim, a Seção 5 conclui o artigo, também discutindo trabalhos futuros.

2. Eficiência energética em centro de dados

Alguns trabalhos têm sido propostos na literatura com o objetivo de promover a redução do consumo de energia em redes de centros de dados. Entre os mais relevantes, podemos citar o *ElasticTree*, o GreenTE, o PCube e o HERO, os quais discutimos a seguir.

O *ElasticTree* [Heller et al., 2006] consiste numa proposta que busca a eficiência energética em redes de centro de dados baseadas na topologia *Fat Tree*. A solução faz uso de três módulos responsáveis por monitorar continuamente as condições de tráfego do centro de dados, selecionar o conjunto de elementos de rede que precisam estar ativos a fim de atender aos requisitos de desempenho e tolerância a falhas, e desligar os *links* e *switches* não envolvidos no roteamento. O GreenTE [Zhang et al., 2010] consiste na criação de um controlador lógico centralizado capaz de manipular caminhos através do OSPF

e criar um túnel MPLS de modo a deixar o menor número possível de roteadores ativos para satisfazer as restrições de desempenho, tais como demandas de tráfego e atrasos de pacotes. A eficiência energética é alcançada quando ele desativa os roteadores e *links* ociosos. O PCube [Huang et al., 2011] propõe melhorar a eficiência energética através do ajuste dinâmico da estrutura de redes baseadas na topologia BCube a partir das demandas de tráfego. A solução faz uso de um gerenciador que fica responsável por analisar e otimizar o tráfego da rede; escalonar os pacotes de dados; e realizar alterações na topologia de rede de forma que fiquem ativos apenas os *links* e equipamentos que estiverem em uso. O HERO (do inglês, *Hierarchical EneRgy Optimization*) visa alcançar eficiência energética em redes hierárquicas [Zhang e Ansari, 2012]. A solução desativa os *links* e equipamentos ociosos através de uma heurística que implementa um algoritmo "guloso".

Embora existam vários trabalhos relacionados ao provimento de eficiência energética em redes de centro de dados, geralmente as propostas não conseguem aliar eficiência energética com redundância e desempenho, possuindo como características: (i) os *switches* da camada de acesso não sofrem quaisquer tipo de alterações específicas; (ii) má utilização da largura de banda e do controle de congestionamento; (iii) alguns cenários contam com um alto valor de perda e atrasos de pacotes causado pelo *overhead* nos roteadores; (iv) como algumas propostas utilizam algoritmos "gulosos", as decisões tomadas são melhores somente em alguns momentos específicos da comunicação da rede.

3. Solução proposta

Este trabalho apresenta a estratégia BEEP (do inglês, *Balanced Energy, rEdundancy, and Performance*), voltada para o gerenciamento de energia para redes de centro de dados baseadas na topologia do tipo *Fat Tree*. Visando alcançar o equilíbrio entre consumo de energia, redundância mínima necessária de equipamentos e o desempenho da rede, a BEEP procura fazer com que o maior número possível de *switches* permaneçam em estado ocioso e que sejam explorados a maior quantidade possível de caminhos distintos entre os nós comunicantes. Para isso, dois algoritmos são utilizados com as seguintes funções: (i) monitorar a utilização dos *switches* da topologia a fim de deixá-los na maior parte do tempo possível em modo de espera ou com suas interfaces desligadas (com base em sua ociosidade); (ii) adequar o MPTCP à quantidade de fluxos simultâneos nas interfaces dos *hosts* comunicantes ao valor mais apropriado para a comunicação; e (iii) colocar as interfaces dos *switches* ociosos em modo de espera ou desligá-las, reativando-as conforme a necessidade.

Os algoritmos propostos são descritos nas próximas subseções. Eles foram concebidos dentro da perspectiva da *Fat Tree* a ser implementada seguindo o modelo de controle e gerenciamento centralizado, obtido por meio das Redes Definidas por Software – SDN (do inglês, *Software-Defined Network*).

3.1. Algoritmo Desativação de *Switches* (DSW)

O algoritmo de Desativação de *Switches* (DSW) — Algoritmo 1 — é executado no controlador da rede. O mesmo tem como função realizar uma monitoração constante dos *switches* da topologia, a fim de detectar a possível ociosidade dos mesmos. Uma vez que cada regra será inserida na *Flow-Table* com o `idle_timeout`[1] definido em 10 segun-

[1]O parâmetro `idle_timeout` define o tempo máximo de ociosidade que uma regra pode ter para continuar na *Flow-Table* do *Switch*.

dos, o DSW considera ocioso aquele *switch* que após este período tenha sua tabela de fluxos vazia. Esta premissa foi adotada por conta da grande e diversificada demanda de tráfego existente em centro de dados e pela baixa probabilidade de existirem tabelas de fluxos sem alterações em *switches* ativos. Logo, os equipamentos identificados pelo DSW no estado ocioso têm suas interfaces desligadas ou colocadas em modo de espera. Este procedimento é realizado a partir do conjunto de *switches* de acesso da topologia (entrada SW_{acesso}) e o conjunto de *switches* de agregação e *core* da topologia (entrada $SW_{FatTree}$).

A partir dos conjuntos de entrada, o DSW cria como saída dois conjuntos de *switches*. No primeiro, são reunidos os *switches* que estão com as interfaces em modo de espera (SW_{espera}), e no segundo, os *switches* que estão com as interfaces desligadas ($SW_{desligados}$). O primeiro passo do DSW é obter a quantidade dos *switches* da topologia (linha 1). Em seguida o DSW passa todos os *switches* para o modo de eficiência energética, a partir da configuração das interfaces dos *switches* de acesso para o modo de espera e desligamento daquelas pertencentes aos *switches* de agregação e *core* (linhas 2 e 3). O trecho entre as linhas de 4 a 16 consiste em um *loop* infinito, no qual a cada tempo δ (10 segundos) são verificadas as tabelas de fluxo de todos os *switches* da topologia. Contudo, como apenas os *switches* tem conexão direta com o controlador, se essas interfaces fossem completamente desligadas, não haveria uma forma de os *switches* da camada de acesso "perceberem" que determinados *hosts* desejariam trafegar dados. Por tais motivos, as interfaces dos *switches* de acesso devem permanecer em modo espera.

Algoritmo 1: Desativação de *Switches* (DSW)

```
   Entrada: SW_acesso, SW_FatTree
   Saída: SW_espera, SW_desligados
1  qtde_switches = n(SW_acesso) + n(SW_FatTree);
2  colocar todos os switches de SW_acesso em modo de espera e incluí-los em
   SW_espera;
3  colocar todas os switches de SW_FatTree em modo desligado e incluí-los em
   SW_desligados;
4  para (cada δ) faça
5      para (i de 1 até qtde_switches) faça
6          qtde_fluxos = quantidade de entradas na Flow Table do switch[i];
7          se (qtde_fluxos = 0) então
8              se (switch[i] ∈ ao conjunto SW_acesso) então
9                  Colocar interfaces de SW_acesso em modo de espera e incluí-lo
                   em SW_espera;
10                 senão
11                     Colocar interfaces de SW_FatTree em modo desligado e
                       incluí-lo em SW_desligados;
12                 fim se
13             fim se
14         fim se
15     fim para
16 fim para
```

3.2. Algoritmo Ativação de *Switches* (ASW)

O algoritmo de Ativação de *Switches* (ASW) — Algoritmo 2 — aproveita a visão global da rede para explorar os múltiplos caminhos existentes entre dois *hosts*. Assim, é possível alocar fluxos de dados em mais de um caminho distinto de forma a utilizar, da melhor maneira possível, toda a largura de banda que a topologia tem a oferecer. A partir da visão global da rede, o ASW não somente maximiza o uso de recursos da infraestrutura como também melhora o desempenho ao agregar seus fluxos em canais com menor tempo de resposta.

Quando há uma solicitação de tráfego, o ASW recebe como entrada os valores referentes aos *hosts* de origem e destino da comunicação, o conjunto de *switches* que estão em modo espera, o conjunto de *switches* que estão em modo desligado (recebidos do Algoritmo DSW) e, por fim, os limiares mínimo e máximo para a quantidade de sub-fluxos por interfaces (variáveis h_{origem}, $h_{destino}$, SW_{espera}, $SW_{desligados}$, α e β, respectivamente). A partir destes parâmetros, o ASW computa o *switch* que conecta o *host* de origem (SW_{origem}), o *switch* que conecta o *host* de destino ($SW_{destino}$), o POD (*Point of Delivery*) do *host* de origem (pod_{origem}) e o POD do *host* de destino ($pod_{destino}$). Em seguida, quatro conjuntos são criados contendo: (i) os *switches* de acesso (SW_{acesso}); (ii) os *switches* de agregação ($SW_{agregacao}$); (iii) os *switches core* (SW_{core}); e (iv) os *switches* que devem ficar ativos durante a comunicação (SW_{ativos}). O ASW, então, adiciona os *switches* de origem e destino no conjunto dos *switches* que devem estar ativos durante a comunicação. Se o *switch* de origem for igual ao *switch* de destino, o ASW adiciona o identificador de apenas um deles no conjunto de *switches* ativos (cláusula *distinct*).

Conforme os *hosts* de origem e destino, o algoritmo ASW se comporta de modo diferente de acordo com as três formas possíveis de comunicação: (i) comunicação entre *hosts* vizinhos conectados ao mesmo *switch* (Comunicação *Intra-Switch*); (ii) comunicação entre *hosts* pertencentes ao mesmo POD (Comunicação *Intra-POD*); e (iii) comunicação entre *hosts* pertencentes a PODs diferentes (Comunicação *Inter-POD*). A Figura 1 ilustra esses três tipos de comunicação em uma rede que segue a topologia *Fat Tree* K4.

Na comunicação *Intra-Switch* existe apenas um único caminho entre dois *hosts* vizinhos (linha 7). Como a vantagem do MPTCP está na possibilidade de envio de dados por caminhos distintos, este é o único caso em que o mesmo não pode ser utilizado. O algoritmo então prossegue pela definição da quantidade de sub-fluxos por interface ($fl_{interface}$) com um valor limiar mínimo α, utilizado para configuração da quantidade de sub-fluxos por interface a partir da linha 20. A Figura 1(a) apresenta um exemplo desse tipo de comunicação em que os *hosts* `h1` e `h2` estão conectados no mesmo *switch* `s13`. Nesta situação, todos os demais *switches* são passíveis de serem desligados pelo DSW.

Na comunicação *Intra-POD* o tráfego ocorre entre *hosts* que, embora estejam interligados por *switches* de acesso diferentes, estão dentro do mesmo POD. Neste caso, como o ASW já incluiu os *switches* de origem e destino no conjunto de *switches* que devem estar ativos (linha 6), ele adiciona também neste conjunto os *switches* de agregação do POD em que está havendo a comunicação. Além disso, o ASW define o valor da variável $fl_{interface}$ com a quantidade destes *switches* de agregação (linha 12). O algoritmo, então, prossegue configurando a quantidade de sub-fluxos por interfaces dos *switches* que estão no conjunto de *switches* ativos e verificando se os mesmos estão em modo de espera ou desligados (linhas 20 a 36). A Figura 1(b) apresenta um exemplo de comunicação

Algoritmo 2: Ativação de *Switches* (ASW)

Entrada: $h_{origem}, h_{destino}, SW_{espera}, SW_{desligados}, \alpha, \beta$

1 $SW_{espera} \Leftarrow$ conjunto dos *switches* que estão em espera (recebido do Algoritmo DSW);
2 $SW_{desligados} \Leftarrow$ conjunto dos *switches* que estão desligados (recebido do Algoritmo DSW);
3 $SW_{ativos} \Leftarrow$ conjunto de *switches* ativos (inicialmente $\emptyset$);
4 sw_{origem} = *switch* de origem baseado em h_{origem}; $sw_{destino}$ = *switch* de destino baseado em $h_{destino}$;
5 pod_{origem} = POD de origem baseado em sw_{origem}; $pod_{destino}$ = POD de destino baseado em $sw_{destino}$;
6 Incluir no conjunto SW_{ativos} os *switches* de SW_{origem} e $SW_{destino}$ (*distinct*);
7 **se** *(*$sw_{origem} = sw_{destino}$*)* **então**
8 $fl_{interface} = \alpha$ /* Há apenas 1 *switch* no caminho de comunicação */;
9 **senão se** *(*$pod_{origem} = pod_{destino}$*)* **então**
10 /* A quantidade de caminhos é igual a quantidade de *switches* de agregação no POD */
11 Incluir no conjunto SW_{ativos} os *switches* de agregação do POD de origem;
12 $fl_{interfaces}$ = quantidade de switches de agregação do POD de origem;
13 **senão**
14 /* A quantidade de caminhos é igual a quantidade de *switches core* da topologia */
15 Incluir no conjunto SW_{ativos} os *switches* de agregação do POD de origem e POD de destino, e os switches core da topologia;
16 $fl_{interfaces}$ = quantidade dos *switches core* da topologia;
17 **fim se**
18 **fim se**
19 **fim se**
20 **se** *(*$fl_{interface} = \alpha$*)* **então**
21 Configura a quantidade de sub-fluxos por interface igual ao limiar mínimo (α);
22 **senão se** *(*$fl_{interface} \leq \beta$*)* **então**
23 Configura a quantidade de sub-fluxos por interface igual ao valor de $fl_{interface}$;
24 **senão**
25 Configura a quantidade de sub-fluxos por interface igual ao limiar máximo (β);
26 **fim se**
27 **fim se**
28 **fim se**
29 **para** *(i de 1 até n(*SW_{ativos}*))* **faça**
30 **se** *(*$SW_{ativos}[i] \in SW_{espera}$*)* **então**
31 Retirar interfaces de $SW_{ativos}[i]$ do modo espera;
32 **fim se**
33 **se** *(*$SW_{ativos}[i] \in SW_{desligados}$*)* **então**
34 Retirar interfaces de $SW_{ativos}[i]$ do modo desligado;
35 **fim se**
36 **fim para**

Intra-POD. Note que a comunicação entre os *hosts* `h1` e `h4` pode acontecer por dois caminhos distintos, dado que os referidos *hosts* são interligados por *switches* diferentes.

Nos casos do tráfego acontecer entre *hosts* que estejam em PODs diferentes, o ASW adiciona também, no conjunto de *switches* ativos, os *switches* de agregação dos PODs de origem e destino e os *switches core* da topologia. Além disso, define o valor dos sub-fluxos com a quantidade destes *switches* core (linhas 20 a 27). A Figura 1(c) apresenta um exemplo da comunicação *Inter-POD*. Neste exemplo, a comunicação entre os *hosts* `h1` e `h16` acontece por *switches* diferentes, localizados em PODs também diferentes. Assim, existem quatro caminhos distintos possíveis em uma eventual comunicação entre

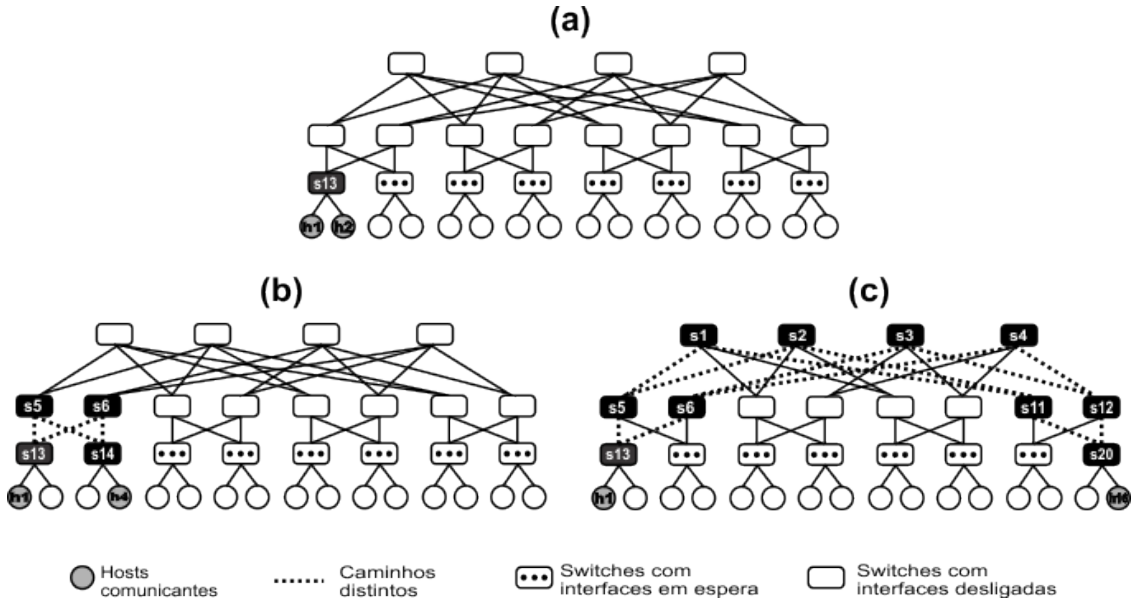


Figura 1. Formas de comunicação na topologia *FatTree*: (a) Intra-Switch; (b) Intra-POD; e (c) Inter-POD.

os *hosts*. Por fim, a partir da linha 23, o ASW configura a quantidade de sub-fluxos por interfaces e verifica se os *switches* que estão no conjunto de *switches* ativos estão em modo de espera ou desligados.

4. Experimentação: emulação e testes realizados

Os algoritmos descritos na Seção 3 foram avaliados através de redes de centro de dados definidas por *software* nas topologias *Fat Tree*, implementadas por meio da plataforma *OpenFlow* e o MPTCP. O *OpenFlow* oferece o controle centralizado da rede, permitindo maior flexibilidade na definição de regras das tabelas de fluxos no plano de encaminhamento formado pelos *switches*, assim como, na identificação dos possíveis caminhos entre *hosts* da *Fat Tree*. Já o MPTCP permite melhor utilização da largura banda disponível, uma vez que possibilita o envio do tráfego por caminhos distintos. Embora o ECMP (do inglês, *Equal-Cost Multi-Path routing*) [Hopps, 2000] consiga enviar o tráfego por mais de um caminho, o seu escalonamento não realiza uma divisão justa da largura de banda. Isso faz com que alguns *links* fiquem sub-utilizados e possuam maior taxa de perdas de pacotes e retransmissões durante a comunicação [Alizadeh et al., 2010].

Nos experimentos realizados, a solução descrita na Seção 3 foi analisada em comparação com uma rede utilizando o ECMP, a partir da média de 50 execuções, conforme [Paasch et al., 2013], com intervalos de confiança no nível de 95%, observando as seguintes métricas: (i) consumo de energia por *switch* (medida em *Watts* / Hora); (ii) *goodput*; (iii) latência; (iv) taxa de perdas; e (vi) quantidade de vezes em que as interfaces dos *switches* ficam em modo de espera ou desligadas.

A adoção do ECMP como referência de comparação deve-se à dificuldade de reproduzir a implementação de outros trabalhos da literatura. Ademais, a estratégia BEEP não pode ser comparada, integralmente, com outras soluções por ter o objetivo de buscar o equilíbrio entre Energia, Redundância e Desempenho.

4.1. Ambiente de testes

As tecnologias utilizadas no ambiente de emulação dos experimentos foram: (i) *PC Core* i5, 2.3*Mhz*, 8*GB* de *RAM*; (ii) Sistema Operacional *GNU/Linux*, *x*64, versão do *kernel* 3.17.2; (iii) *MultiPath* TCP em *Kernel* do *Linux*[2], versão estável 0.89.2; (iv) *Mininet*[3], versão 2.1.0p1; (v) *Switch Virtual OpenFlow (Open vSwitch)*[4], versão 2.0.2; (vi) Protocolo *OpenFlow*[5], versão 1.0; (vii) Controlador *OpenFlow* RYU[6], versão 3.15; e (viii) Gerador de tráfego D-ITG[7], versão 2.4.6. A escolha por estas ferramentas deveu-se à ampla documentação existente e à facilidade de implementação.

O desenvolvimento das funcionalidades do controlador RYU envolveu os seguintes módulos, que são executados simultaneamente: (i) **Switch MPTCP** — configurado para executar o MPTCP. Este módulo insere, mede e coleta indicadores da utilização dos *switches OpenFlow* da rede. Ao ser executado, esse módulo reduz a velocidade de todas as interfaces dos *switches* da topologia passando-as de 10Gbps para 1Gbps nos *switches* da camada *core* e de 1Gbps para 100Mbps os demais; (ii) **DSW** — desenvolvido para executar o algoritmo DSW; (iii) **ASW** — desenvolvido para executar o algoritmo ASW.

4.1.1. Configuração do MPTCP

O MPTCP foi configurado com o gerenciador de caminhos `ndiffports`. Este gerenciador foi projetado especificamente para utilização em redes de centros de dados com a função de explorar seus vários caminhos distintos [Paasch et al., 2013]. Essa configuração permite que as interfaces de rede possuam mais de um fluxo.

A possibilidade de criar mais de um fluxo por interface pode fazer o MPTCP perder o seu desempenho devido a problemas com a fragmentação do protocolo IP. A quantidade de fluxos muito elevada pode resultar em erros no cálculo do MTU. Por tais motivos, a fim de evitar tais perdas de desempenho, foram configurados um limiar mínimo e um máximo para a quantidade máxima de sub-fluxos por interfaces. Os limiares mínimo e máximo são passados para o controlador através das variáveis α (com valor 1) e β (com valor 5), definidas no algoritmo ASW. O limiar mínimo é utilizado quando há apenas um caminho para a conexão, caso contrário, de acordo com a origem e o destino da comunicação, o valor máximo pode ser até 5. Independente da topologia, se a quantidade de sub-fluxos por interface for maior que 5, o MPTCP apresenta perda de desempenho por conta do *overhead* gerado nas interfaces dos *hosts* comunicantes.

A partir da informação sobre o par de *hosts* envolvidos na determinada comunicação, os sub-fluxos por interfaces são configurados como:

- **Comunicação *Intra-Switch***: nesta comunicação só há um caminho possível. Por este motivo estas conexões utilizam o limiar mínimo para a quantidade de sub-fluxos por interface que tem o valor 1;

[2]http://multipath-tcp.org
[3]http://mininet.org/
[4]http://openvswitch.org/
[5]http://archive.openflow.org/
[6]http://osrg.github.io/ryu/
[7]http://traffic.comics.unina.it/software/ITG/

- **Comunicação *Intra-POD***: nesta comunicação há mais de um caminho distinto possível. Assim, o valor do limiar será configurado a partir da quantidade de *switches* de agregação presentes no POD em que há a comunicação. Se a quantidade de *switches* de agregação for menor ou igual ao limiar máximo (adotado no experimento como 5), a quantidade de sub-fluxos por interfaces será igual à quantidade de *switches* de agregação que o POD possui. Caso contrário, independente da quantidade de *switches* de agregação que o POD possua, o limiar é configurado com o valor máximo;
- **Comunicação *Inter-POD***: nesta comunicação também há mais de um caminho distinto. Assim, se a quantidade de *switches core* for menor ou igual ao limiar máximo, o valor da quantidade de sub-fluxos a ser configurado nas interfaces será igual a quantidade de *switches core* presentes na topologia. Caso contrário, independente da quantidade de *switches core* que a topologia possua, o valor será configurado com o valor máximo.

Outro aspecto relevante na configuração do MPTCP consiste na escolha do esquema de controle de congestionamento. Neste trabalho, foi implementado o esquema de controle de congestionamento *wVegas* [Cao et al., 2012]. Trata-se de um esquema baseado no tempo de atraso que possui um balanceador de carga melhorado. O *wVegas* consegue promover o "*princípio da igualdade no congestionamento*", em que o MPTCP escalona os pacotes nos caminhos com menor tempo de resposta. O *wVegas* também tem a vantagem de fazer com que até os *links* considerados mais lentos consigam obter um bom desempenho através de seu algoritmo iterativo.

4.1.2. Configuração dos *Switches*

O consumo de energia de um *switch* é dividido entre os seus quatro principais componentes. São eles: o *chassis*, o *switch-fabric*, as *linecards* e as portas. O *chassi* é responsável pelo sistema de refrigeração, dentre outros. O *switch-fabric* é responsável por aprender os MACs e/ou IPs das interfaces e manter as tabelas lógicas do equipamento. A *linecard* é responsável por encaminhar os pacotes do *switch-fabric* para as portas e as portas são responsáveis por realizar as comunicações na rede.

Como a *Fat Tree* opera em uma arquitetura em camada de 3 níveis, geralmente são utilizadas soluções de *switches* com diferentes velocidades. Frequentemente é adotada a velocidade de 10Gbps para *switches* da camada *core* e 1Gbps para os *switches* das camadas de agregação e acesso. Entretanto, como a solução proposta visa reduzir estas velocidades nas interfaces, o consumo dos *switches* de 100Mbps também foi considerado na avaliação. Os valores de consumo utilizados nos experimentos, descritos na Tabela 1, foram adotados de acordo com a linha de *switches OpenFlow* da HP [Packard, 2014].

O modo de espera realizado nas interfaces implementa o WoP (*Wake on Packet*) [Ananthanarayanan e Katz, 2008]. Neste modo, as interfaces ficam em estado de espera e só são reativadas quando há uma solicitação de tráfego. Esse pacote "reativador" da interface se assemelha ao pacote mágico utilizado pelo WoL (*Wake on Lan*), no qual equipamentos são retirados do modo de espera através do processamento de sua interface de rede. No WoP o *switch* deixa a interface consumindo apenas o mínimo possível para ser reativada quando houver necessidade.

Tabela 1. Parâmetros dos *Switches OpenFlow* utilizados nos experimentos.

Parâmetro	Descrição	Valores configurados nos *switches*		
		100Mbps	1Gbps	10Gbps
$Consumo_{maximo}$	Consumo com carga total	120W	130W	685W
$Consumo_{ocioso}$	Consumo no estado ocioso	98W	180W	320W
$Consumo_{porta-ativa}$	Consumo de cada porta ativa	5,4W	8W	14W
$Consumo_{porta-ociosa}$	Consumo de cada porta ociosa	3W	5W	9W
$Consumo_{porta-espera}$	Consumo de cada porta em espera	1,6W	3W	3W
$Consumo_{porta-shutdown}$	Consumo de cada porta desligada	0,1W	0,5W	0,5W
$Tempo_{shutdown-on}$	Tempo para colocar em *shutdown*	0,67ms	1ms	1ms
$Tempo_{shutdown-off}$	Tempo para retirar de *shutdown*	0,88ms	1,2ms	1,4ms
$Tempo_{espera-on}$	Tempo para colocar em modo espera	1ms	1ms	1ms
$Tempo_{espera-off}$	Tempo para retirar do modo espera	1ms	1,1ms	1,1ms
$Tempo_{mudanca-velocidade}$	Tempo para mudança de velocidade	1,2ms	1,3ms	1ms

4.1.3. Tráfego caracterizado

Conforme relatado em [Bilal et al., 2014, Benson et al., 2010], o tráfego TCP é o mais adequado para testes e avaliação de redes de centro de dados que tem por objetivo obter eficiência energética. Por tal motivo, neste trabalho foi utilizado tráfego TCP para os padrões de comunicação um-para-todos e todos-para-todos. Assim, foram geradas cargas de trabalho sintéticas, através do D-ITG (do inglês, *Distributed Internet Traffic Generator*) que se baseia na distribuição *Poisson*. No envio da carga de trabalho, o gerador de tráfego emite fluxos dos tipos "*elephant*", "*tortuoise*" e "*cheetars*".

Por fim, é preciso destacar que nos experimentos o tráfego ficou mais concentrado na camada *core*, com perdas mais freqüentes nas bordas e com 80% dentro do *rack*. Tais parametrizações tornam a geração de tráfego mais próxima de uma rede real e visam permitir que a solução proposta neste artigo seja comparável com qualquer cenários realistas de utilização [Benson et al., 2010, Chen et al., 2011].

4.2. Resultados obtidos

Nos experimentos foram utilizadas sete variantes da topologia *Fat Tree*, emuladas no *Mininet* de acordo com a descrição apresentada na Tabela 2. As velocidades dos *switches* foram configuradas com 10 Gbps para os *switches* da camada *core*, 1 Gbps para *switches* da camada de agregação e 1 Gbps para os *switches* da camada de acesso. A fim de avaliar a solução proposta em tais redes, foram realizados testes através da geração de tráfego com cargas de 1 GB, 5 GB, 10 GB e 20 GB, distribuídos entre os *hosts*.

Os resultados dos experimentos estão apresentados nas Tabelas 3, 4, 5 e 6. As colunas com a sigla ECMP se referem aos resultados obtidos sem qualquer solução para eficiência energética, ou seja, apenas com o ECMP. As colunas com BEEP se referem aos valores obtidos com a estratégia proposta neste trabalho. A coluna "Consumo dos *switches*" se refere a soma do consumo (em *Watts* / Hora), valendo salientar que: (i) quando os valores se referem ao ECMP, os *switches core* operam com velocidade de 10 Gbps, e os demais *switches* a 1 Gbps; (ii) quando os valores se referem a estratégia BEEP, os *switches core* operam com velocidade de 1 Gbps, e os demais *switches* a 100 Mbps.

Com base na Tabela 3, é possível notar que no quesito tempo de transmissão, a cada vez que a topologia é aumentada, o tempo de transmissão com a estratégia BEEP é diminuído. Esse fato se deve à melhor utilização da largura de banda e ao escalonamento

Tabela 2. Topologias *Fat Tree* emuladas nos experimentos.

Topologia	Qtde. de *Hosts*	Qtde. de *Switches*
K4	16	20
K5	20	31
K6	54	45
K7	63	61
K8	128	80
K9	144	101
K10	250	125

entre os vários caminhos distintos realizado pelo MPTCP. Dessa forma, quanto mais *switches* a topologia possuir, maior será o número de caminhos distintos que ela pode oferecer ao MPTCP, o que torna a estratégia, a partir da topologia K6, mais veloz que a solução convencional, mesmo operando com velocidade inferior. Outra questão a destacar é que à medida que a topologia cresce, diminui o percentual de economia de energia obtido com a BEEP. Os resultados também comprovam outra vantagem da estratégia com relação ao consumo médio por *switch*, *Goodput* e as perdas de pacotes. Nos experimentos, em nenhuma situação ele obteve valor pior que a infraestrutura operando apenas com o ECMP. Este ganho se deve ao melhor balanceamento de carga e gerenciamento de congestionamento realizado pelo MPTCP. Por fim, ressaltamos que por conta do maior tempo gasto para o tráfego da carga de trabalho de 1GB na topologia ter sido 6,1 segundos, o algoritmo DSW (Algoritmo 1) não colocou em modo de espera ou desligou nenhuma das interfaces dos *switches*.

Tabela 3. Experimento com carga de 1GB.

1GB de tráfego distribuído aleatoriamente entre os *hosts*														
Topologias	Tempo gasto para transferência (segundos)			Consumo dos *Switches* (*Watts* / Hora)			Consumo médio por *Switch* (*Watts* / Hora)		*Goodput (%)*		Perdas (%)		Interfaces (BEEP)	
	ECMP	BEEP	Diferença	ECMP	BEEP	Economia (%)	ECMP	BEEP	ECMP	BEEP	ECMP	BEEP	Desl.	Espera
K4	6,1	8,3	2,2	4.129,0	2.283,0	44,7	206,45	114,15	89	93	11	9	0	0
K5	5,6	5,7	0,1	6.421,0	3.884,0	39,5	321,05	194,20	89	94	10	8	0	0
K6	4,3	4,0	-0,3	8.427,0	5.128,0	39,1	187,27	113,96	88	91	12	7	0	0
K7	3,2	2,7	-0,5	12.111,0	7.665,0	36,7	198,54	125,66	88	93	13	8	0	0
K8	2,8	2,5	-0,3	15.489,0	10.123,0	34,6	193,61	126,54	88	93	11	7	0	0
K9	2,2	1,8	-0,4	21.321,0	14.889,0	30,2	211,10	147,42	89	94	10	8	0	0
K10	1,5	0,9	-0,6	27.931,0	19.376,0	30,6	223,45	155,01	89	92	11	8	0	0

Com base na Tabela 4, é possível observar que análogo aos resultados dos experimentos com a carga de trabalho de 1GB, a estratégia BEEP mostrou resultados melhores nos quesitos tempo de transmissão, consumo médio por *switch*, taxa de *goodput* e perdas. Como o maior tempo gasto para realizar o tráfego da carga de 5GB foi de 16,2 segundos, apenas 1 *switch* foi desligado e 1 *switch* entrou em modo de espera durante os experimentos nas topologias K4 e K5.

Na Tabela 5, é possível notar que assim como nos experimentos com as cargas anteriores, a solução com a estratégia BEEP continua melhor em relação a infraestrutura com o ECMP. É interessante destacar que o percentual de economia na topologia K5, que contém 31 *switches*, foi maior do que na topologia K4, que contém 20 *switches* (fato justificado pelo número maior de interfaces desligadas pelo Algoritmo DSW na topologia K5).

Com base na Tabela 6, nota-se que o ganho com a estratégia BEEP tem com-

Tabela 4. Experimento com carga de 5GB.

5GB de tráfego distribuído aleatoriamente entre os *hosts*														
Topologias	Tempo gasto para transferência (segundos)			Consumo dos *Switches* (*Watts* / Hora)			Consumo médio por *Switch* (*Watts* / Hora)		*Goodput (%)*		Perdas (%)		Interfaces (BEEP)	
	ECMP	BEEP	Diferença	ECMP	BEEP	Economia (%)	ECMP	BEEP	ECMP	BEEP	ECMP	BEEP	Desl.	Espera
K4	16,2	19,9	3,7	5.384,0	3.476,0	35,4	269,20	173,80	88	91	12	91	1	1
K5	13,8	14,4	0,6	8.003,0	5.008,0	37,4	400,15	250,40	88	92	13	92	1	1
K6	12,9	12,7	-0,2	10.120,0	6.139,0	39,3	224,89	136,42	89	92	15	92	0	0
K7	10,7	9,7	-1,0	13.743,0	9.039,0	34,2	225,30	148,18	89	93	15	93	0	0
K8	9,4	9,0	-0,4	17.347,0	11.075,0	36,2	216,84	138,44	86	94	16	94	0	0
K9	8,3	7,5	-0,8	23.876,0	15.627,0	34,5	236,40	154,72	85	91	15	91	0	0
K10	7,0	5,9	-1,1	29.654,0	19.221,0	35,2	237,23	153,77	85	91	15	91	0	0

Tabela 5. Experimento com carga de 10GB.

10GB de tráfego distribuído aleatoriamente entre os *hosts*														
Topologias	Tempo gasto para transferência (segundos)			Consumo dos *Switches* (*Watts* / Hora)			Consumo médio por Switch (*Watts* / Hora)		*Goodput (%)*		Perdas (%)		Interfaces (BEEP)	
	ECMP	BEEP	Diferença	ECMP	BEEP	Economia (%)	ECMP	BEEP	ECMP	BEEP	ECMP	BEEP	Desl.	Espera
K4	33,0	41,0	8,0	6.876,0	4.228,0	38,5	343,80	211,40	88	91	10	7	4	3
K5	31,8	38,0	6,2	9.327,0	5.550,0	40,5	466,35	277,50	87	90	9	8	6	1
K6	29,5	29,0	-0,5	11.298,0	7.332,0	35,1	251,07	162,93	89	92	12	9	4	1
K7	27,6	24,3	-3,3	14.498,0	9.418,0	35,0	237,67	154,39	87	94	14	9	3	0
K8	26,8	21,8	-5,0	19.287,0	12.511,0	35,1	241,09	156,39	88	92	14	8	1	0
K9	25,2	19,5	-5,7	25.554,0	17.038,0	33,3	253,01	168,69	88	92	16	8	2	0
K10	20,1	14,8	-5,3	30.754,0	20.287,0	34,0	246,03	162,30	88	93	16	7	1	0

portamento similar com as demais cargas, exceto por um detalhe: nas cargas emuladas anteriormente, em que o MPTCP começa a ser mais veloz que a infraestrutura convencional da topologia a partir da topologia K6 em diante, quando a carga foi de 20GB. A topologia K6 com o MPTCP obteve latência um pouco maior que a infraestrutura com apenas o ECMP, o que leva a crer que quanto maior for a carga de trabalho, a latência pode se tornar maior com a estratégia, dada a diferença entre as velocidades das soluções.

Como neste experimento a carga de trabalho foi maior do que as demais e, consequentemente, o seu tempo gasto para transmissão foi maior, ficou mais evidente a importância do algoritmo DSW. O mesmo foi capaz de desligar ou colocar em espera os *switches* que ficaram ociosos durante o referido experimento.

Tabela 6. Experimento com carga de 20GB.

20GB de tráfego distribuído aleatoriamente entre os *hosts*														
Topologias	Tempo gasto para transferência (segundos)			Consumo dos *Switches* (*Watts* / Hora)			Consumo médio por Switch (*Watts* / Hora)		*Goodput (%)*		Perdas (%)		Interfaces (BEEP)	
	ECMP	BEEP	Diferença	ECMP	BEEP	Economia (%)	ECMP	BEEP	ECMP	BEEP	ECMP	BEEP	Desl.	Espera
K4	55,0	72,0	17,0	7.438,0	4.611,0	38,0	371,90	230,55	89	93	11	7	12	4
K5	54,3	66,0	11,7	10.180,0	6.343,0	37,7	509,00	317,15	89	92	13	8	8	2
K6	53,8	58,0	4,2	12.132,0	8.028,0	33,8	269,60	178,40	88	92	14	8	6	3
K7	50,0	46,3	-3,7	16.268,0	10.665,0	34,4	266,69	174,84	88	93	15	8	3	3
K8	48,8	45,1	-3,7	21.568,0	13.697,0	36,5	269,60	171,21	87	92	12	9	3	1
K9	46,9	42,1	-4,8	27.052,0	18.033,0	33,3	267,84	178,54	87	93	14	9	3	0
K10	44,7	39,5	-5,2	33.700,0	23.128,0	31,4	269,60	185,02	85	92	14	9	2	1

4.3. Avaliação geral dos resultados obtidos

Com base nos resultados obtidos, destacamos que: (i) devido à proposta utilizar o maior número possível de equipamentos (caminhos) durante a transmissão a redundância das conexões é mantida; (ii) como o MPTCP utiliza melhor a largura de banda (principalmente por distribuir o tráfego em caminhos menos congestionados) ele possui menor

número de perdas e retransmissões, o que torna as comunicações rápidas, seguras e com melhor consumo energético. Uma vez que os *switches* trafegam mais rápido, ficam mais tempo ociosos, o que os tornam passíveis de entrarem em modo de eficiência energética); e (iii) quanto maior for o número de caminhos distintos existente entre o par origem e destino, melhor é o desempenho do MPTCP.

Dado o exposto, ressaltamos que a utilização da estratégia BEEP traz benefícios quando utilizada em redes de centro de dados, sendo que sua utilização proporciona um equilíbrio entre o consumo de energia e a redundância da topologia, além de aumentar o desempenho das conexões.

5. Conclusão e trabalhos futuros

Com o aumento da escala de centro de dados, os sistemas de gestão de energia centralizados para redução do consumo de energia nos equipamentos destas redes possuem problemas de escalabilidade. Com base na topologia hierárquicas *Fat Tree* e padrões de tráfego de centro de dados, propusemos e avaliamos a solução para redução do consumo de energia através da utilização de SDN, *OpenFlow* e o *MultiPath* TCP sem violar as restrições de conectividade e de redundância.

Através dos resultados experimentais obtidos por emulação (Seção 4.2), foi verificado que a estratégia BEEP mostra-se competitiva diante das soluções atualmente existentes para infraestrutura de redes de centro de dados. Através da exploração do controle centralizado obtido pelo controlador *OpenFlow*, aliado ao desempenho oferecido pelo MPTCP e pelo seu modo de escalonar e controlar congestionamentos, a solução proposta consegue aliar eficiência energética com melhor distribuição de banda e recursos.

Para trabalhos futuros é necessário fazer um estudo com uma gama maior de cargas e de topologias para centro de dados, realizar testes de escabilidade, testar o melhor cenário para aplicação de regras pró-ativas e reativas, bem como emular o tráfego mais próximo possível de ambientes de produção. Uma outra forma de avaliar melhor a solução seria implementar um mecanismo capaz de não só reduzir as velocidades das interfaces, mas fazer estes ajustes de acordo com a caracterização do tráfego e volume de carga de trabalho nos *switches* em comunicações originadas dentro e fora do centro de dados.

Agradecimentos

Os autores agradecem o apoio da CAPES, CNPq e FAPERJ.

Referências

Alizadeh, M., Greenberg, A., Maltz, D. a., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., e Sridharan, M. (2010). Data center TCP (DCTCP). *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM '10*, page 63.

Ananthanarayanan, G. e Katz, R. (2008). Greening the Switch. *HotPower. Pages 1-5*.

Barré, S., Paasch, C., e Bonaventure, O. (2011). Multipath tcp: From theory to practice. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, NETWORKING'11, pages 444–457, Berlin, Heidelberg. Springer-Verlag.

Benson, T., Anand, A., Akella, A., e Zhang, M. (2010). Understanding data center traffic characteristics. *SIGCOMM Comput. Commun. Rev.*, 40(1):92–99.

Bilal, K., Malik, S. U. R., Khalid, O., Hameed, A., Alvarez, E., Wijaysekara, V., Irfan, R., Shrestha, S., Dwivedy, D., Ali, M., Shahid Khan, U., Abbas, A., Jalil, N., e Khan, S. U. (2014). A taxonomy and survey on Green Data Center Networks. *Future Generation Computer Systems*, 36:189–208.

Cao, Y., Xu, M., e Fu, X. (2012). Delay-based congestion control for Multipath TCP. *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–10.

Chen, K., Hu, C., Zhang, X., Zheng, K., Chen, Y., e Vasilakos, A. (2011). Survey on routing in data centers: insights and future directions. *Network, IEEE*, 25(4):6–10.

Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., e Sengupta, S. (2009). Vl2: A scalable and flexible data center network. *SIGCOMM Comput. Commun. Rev.*, 39(4).

Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., e Lu, S. (2009). Bcube: A high performance, server-centric network architecture for modular data centers. *SIGCOMM Comput. Commun. Rev.*

Hammadi, A. e Mhamdi, L. (2014). A survey on architectures and energy efficiency in Data Center Networks. *Computer Communications*, 40:1–21.

Heller, B., Seetharaman, S., e Mahadevan, P. (2006). ElasticTree : Saving Energy in Data Center Networks. *Energy*, page 17.

Hopps, C. (2000). Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992 (Informational).

Huang, L., Jia, Q., Wang, X., Yang, S., e Li, B. (2011). PCube: Improving power efficiency in data center networks. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 65–72.

IEEE (2014). IEEE P802.3az Energy Efficient Ethernet Task Force. Disponível em: www.ieee802.org/3/az/. Acessado em 06/11/2014.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., e Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Nordman, B. (2007). Energy Efficient Ethernet: Outstanding Questions. *IEEE 802 interim meeting*, 1(510):1–10.

Paasch, C., Khalili, R., e Bonaventure, O. (2013). On the benefits of applying experimental design to improve Multipath TCP. *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies - CoNEXT '13*, pages 393–398.

Packard, H. H. (2014). Software-defined networking. Disponível em: www8.hp.com/us/en/networking/solutions/technology/sdn/. Acessado em 23/10/14.

Zhang, M., Yi, C., Liu, B., e Zhang, B. (2010). Greente: Power-aware traffic engineering. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 21–30.

Zhang, Y. e Ansari, N. (2012). Hero: Hierarchical energy optimization for data center networks. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2924–2928.

Managing Access to Service Providers in Federated Identity Environments: A Case Study in a Cloud Storage Service

Thomas Diniz[1], **Andre Castro de Felipe**[1], **Taina Medeiros**[1], **Carlos Eduardo da Silva**[1], **Roberto Araujo**[2]

[1]Metropole Digital Institute – Federal University of Rio Grande do Norte (UFRN)
Av. Senador Salgado Filho, S/N – 59.098-970 – Natal, RN, Brazil

[2]Faculdade de Computação – Universidade Federal do Pará (UFPA)
Rua Augusto Corrêa, 01 – 66.075-110 – Belém, PA, Brazil

{thomasfdsdiniz,andrecastroit,tainajmedeiros}@gmail.com

kaduardo@imd.ufrn.br, rsa@ufpa.br

Abstract. *Currently the diversity of services, which are adhering to Identity Federation, has raised new challenges in the area. Increasingly, service providers need to control the access to their resources by users from the federation as, even though the user is authenticated by the federation, its access to resources cannot be taken for granted. Each Service Provider (SP) of a federation implements their own access control mechanism. Moreover, SPs might need to allow different access control granularity. For instance, all users from a particular Identity Provider (IdP) may access the resources due to some financial agreement. On the other hand, it might be the case that only specific users, or groups of users, have access to the resources. This paper proposes a solution to this problem through a hierarchical authorization system. Our approach, which can be customized to different SPs, allows the SP administrator to manage which IdPs, or users, have access to the provided resources. In order to demonstrate the feasibility of our approach, we present a case study in the context of a cloud storage solution.*

1. Introduction

Identity management consists of an integrated system of policies, technologies and business processes that allow organizations the treatment and handling of the identities (identity attributes) of its members [Jøsang and Pope 2005]. From the perspective of a service provider (SP), which makes services and resources available through the Internet, identity management allows an SP to know who its users are (by means of authentication) and to manage what services they are entitle to use (by means of authorization) [Hu et al. 2014].

Different identity management models have been proposed for dealing with, for example, authentication related issues (e.g. [Bhargav-Spantzel et al. 2007, Jøsang et al. 2005]). One example is the federated identity management (FIM) model, where different providers form an association, establishing a trust relationship between them [Chadwick 2009]. In this model, *Identity Providers* (IdP) are responsible for authenticating users, sending messages containing authentication and authorization credentials

of users to *Service Providers* (SP). In this way, a user can access resources offered by different SPs using a single set of credentials issued by an IdP of the federation.

FIM is a model that has been increasingly used in real world scenarios. One of these scenarios is the Academia. Different universities and research centres are increasingly providing some sort of service that can be consumed by users from different institutions. This scenario has called for an academic FIM environment, which is often characterized by the term "academic federation" [Moreira et al. 2014]. In this way, each institution that takes part in the academic federation provides an IdP for authenticating and emitting credentials of its users, while SPs provide their services to users linked to these institutions. One such academic federation, denominated CAFe (*Comunidade Acadêmica Federada*), is currently maintained by the Brazilian NREN (RNP - *Rede Nacional de Ensino e Pesquisa*). Besides the CAFe federation, RNP also offers a series of services to the Brazilian academic community. Other well known federations are: Chimarrão and CAFe Expresso. Chimarrão is a federation responsible for approval IdPs and SPs candidates to be part of CAFe. Since the CAFe Expresso is a federation for experimental purposes, in a way that new technologies and scenarios are tested on it to be matured and after adhered by the Chimarrão and CAFe federations. SE-CNC has users coming from those three federation.

Although IdPs provide authentication, SP administrators very often intend to control whose users access their resources, and with different access control granularity. This is the case of SE-CNC (Experimental Service - Cloud Computing for Science), an experimental cloud service deployed by RNP. SE-CNC is a cloud storage service for the Brazilian academic community to be offered through the CAFe federation. Because SE-CNC is an experimental service, only users from a few institutions of the federation may have access to the service at the moment, characterizing a scenario where all users from a particular IdP may have access to the resources. On the other hand, we also have the case where only specific users, or groups of users, have access to the service. Moreover, due to its community deployment model, several institutions could collaborate to increase the cloud capacity, which could be reflected on different privileges granted to the users associated with those institutions (e.g., an increased storage quota).

These scenarios describe a problem that can be faced by any SP that joins an identity federation: how to delegate access control policy definition of a particular service to IdP administrators. Such delegation can be used, for example, in a scenario where an institution has a total quota in the cloud storage service that can be distributed to its users according to its own rules, constrained by the total quota allocated to the institution.

This paper proposes a solution to this problem through a hierarchical authorization system, which can be customized to different SPs. Our approach trusts the authentication provided by IdPs of the federation, allowing an SP administrator to manage which IdPs, or users, have access to the provided resources, and supporting the delegation of the management of users of a particular IdP to its administrator. In order to demonstrate the feasibility of our approach, we present a case study in the context of the SE-CNC experimental cloud storage service.

The remain of this paper is organized as follows. Section 2 contextualizes the paper, presenting the SE-CNC cloud storage service, the different scenarios that moti-

vated our approach, and a brief background on some concepts used throughout the paper. Section 3 describes our approach, while Section 4 details its implementation. Section 5 discuss some related work found in the literature. Section 6 concludes the paper and present some future directions.

2. Contextualization

This section introduces the context in which our solution has been applied, describing the SE-CNC cloud storage service. In the sequence, we present a brief discussion on the user management scenarios that motivated our approach. Finally, we present the main concepts related with access control that have been considered by our work.

2.1. SE-CNC Cloud Storage Experimental Service

SE-CNC[1] is an experimental cloud storage service sponsored by the Brazilian NREN RNP (Rede Nacional de Ensino e Pesquisa). Its aim is to offer a cloud storage service for professors and researchers in Brazil. In order to accomplish this, it provides functionalities similar to known cloud services such as Google Drive, OneDrive, and Dropbox. For instance, it allows users to easily store and share data using computers or mobile devices.

As a differential from other solutions, SE-CNC intends to provide a more secure data storage service. This is achieved by employing cryptography to encrypt the stored data, allowing encryption at both the server side and the client side, and by locating all of its servers within the RNP infrastructure in Brazilian territory, being subject to Brazilian laws.

The SE-CNC cloud follows the IaaS and SaaS models defined by NIST [Mell and Grance 2011]. That is, it has a physical infrastructure composed of servers that support the service as well as it disposes of a friendly environment to its users. In its experimental phase, the service is deployed as a community model. It is planned to be used by the community of RNP users that includes federal universities and research institutes, which contribute with resources to increase the cloud service capacity.

A general view of the SE-CNC cloud service is presented in Figure 1. The cloud service is composed of a frontend and a backend. The frontend is the part of the service that the users interact with by means of three different types of clients: a Web based client, a desktop synchronization client, and a mobile client. Differently, the backend stores users data. It includes all software necessary for storing data, managing and monitoring the cloud. Both the frontend and the backend are based on open source software.

The solution uses as its front end the ownCloud[2] community edition software. ownCloud acts as a middleware between users and the back end. It authenticates users by means of an identity federation, and control the access to the cloud resources. Through this tool, users can store their data in the cloud. In addition, it allows users to collaborate with others by sharing their data. The CNC cloud service runs in two Web servers geographically away from each other, accessed by a load balancer.

As back end, the cloud storage uses the OpenStack[3] software. In particular, the Swift object storage. The Swift is the tool responsible for storing users data. It ensures

[1] `https://cnc.rnp.br/`
[2] `https://owncloud.org/`
[3] `https://openstack.org`

Figure 1. General View of the SE-CNC Cloud Infrastructure.

the consistency and the availability of the stored data. The Swift tool has two main components: the proxy and the storage services. The proxy service receives all requests to store or obtain data. The storage service stores data and replicates it.

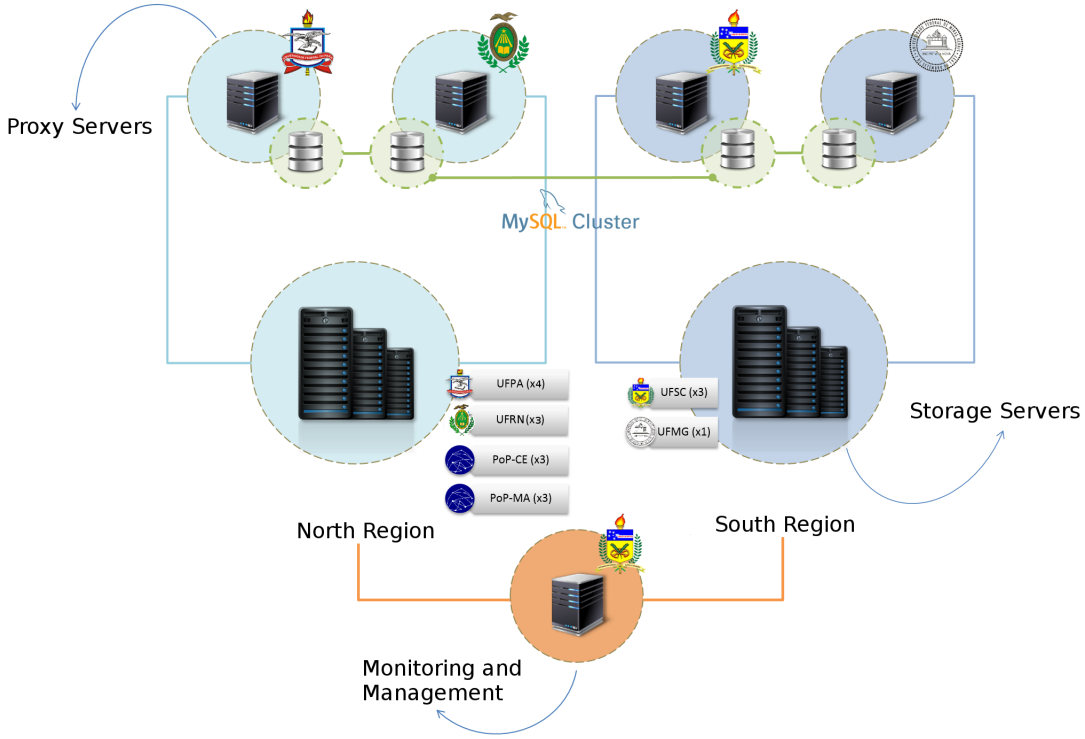


Figure 2. CNC cloud infrastructure.

Figure 2 illustrates the current infrastructure of the CNC cloud. Our infrastructure contains four proxy servers located in federal universities throughout Brazil: federal universities of Pará (UFPA), Rio Grande do Norte (UFRN), Minas Gerais (UFMG), and Santa Catarina (UFSC). Regarding storage servers, the CNC cloud infrastructure contains 15 storage servers distributed in the following way: four servers at UFPA; three servers at UFRN; three servers at UFSC; one server at UFMG; three servers at the RNP Point of Presence of Maranhão (*PoP-MA*) and Ceara (*PoP-CE*).

2.2. The User Management Scenarios

In its experimental phase, users from 10 institutions have accessed the SE-CNC cloud service. They have been using the service as they see fit, and asked to follow some

specific test procedures defined by development team. The users then provide feedback on the tests conducted on a monthly base, and on any other problem that may rise during their daily use.

These interactions with the service clients, and with RNP, have raised two issues related to access control by the SP. The first issue is related to which users (and from which institutions), can access the service. The second issue concerns different access privileges for these users, which is usually captured by means of access control policies specific to the service, i.e., the quota each user has in the storage service.

Concerning the first issue, "the right to access the service", we have identified three different scenarios. Bear in mind that, in these three scenarios, we are still not considering the different privileges users may have within the service.

The first scenario is when we add an SP to the federation, and all users of that federation have access to the service through their respective IdPs. This is the simplest scenario, as the process of joining a federation is enough to allow its users to access the service.

The second scenario is when only users of some IdPs of the federation have access to the service being offered. This is the case when the use of a particular service requires some kind of contractual obligation between the institution offering the service and the institution that manages the IdP. In this case, once the IdP has been "authorized" by the SP, all IdP users will have access to the service. Notice that in this scenario, the trust relationship established by the academic federation is used to provide authentication through IdPs, while the authorization is still controlled by the SP.

The third scenario involves a finer grain access control, where only some users of the federation have access to the SP. This could be the case where the resources offered by the SP are restricted, and either the SP requires some sort of user agreement to be signed by each user, or only specific users of an IdP may have access to the service, as defined by, for example, the institution that manages the IdP.

The second and third scenarios capture the situation dealt with by the SE-CNC cloud storage service. As the service is currently following a community deployment model, it is expected that institutions contribute with the cloud infrastructure (i.e., with the provision of servers), while the users of those collaborating institutions will have access to the cloud service, reflecting the situation presented in the second scenario. However, the cloud service is still in an experimental phase and thus, only a few users from selected client institutions (10 users for each one of the 10 institutions), has access to the service at the moment. As the choice of which user access the service is made by each participant client institution, there is a need to allow some sort of hierarchical access control mechanism, allowing each client institution to appoint an administrator to manage its user's institution access, delegated by the SE-CNC cloud service administrator. This not only would reduce the burden of the SP administrator, but it also allows more control in the distribution of quotas per users.

Besides granting access to users, a SP still needs to manage the different privileges users have within the service. These access control policies are usually SP specific in the sense that the SP knows what actions can be performed within the service it provides. We consider the situation where users may have different access levels to the service

according to rules established by their home institution, i.e., the institution that manages the IdP. Considering the SE-CNC cloud as an example, one particular case that has been raised involves a particular quota allocated to an university (say 100TB), which can then be allocated to the members of the university. For example, all students from a particular department of an university may have a quota of 2 GB, while technical staff may have a quota of 5GB. Since the IdP is managed by an university administrator, we believe that the policies regarding quota should be defined by this administrator, as long as the quota allocated to the University (i.e., the total quota) is not violated.

2.3. A Background on ABAC/RBAC

According to [Sandhu et al. 1996] the central notion of the RBAC (Role-Based Access Control) model is that permissions are associated with roles, and users are assigned to appropriate roles. In this way, an user has access to resources based on the permissions granted to his roles. For example, a subject associated to role Developer has access to a different set of objects than someone associated to role Engineer. Once a subject requests access to an object, the RBAC engine computes which operations are associated to the given role attributed to this subject.

On the other hand, ABAC (Attribute-Based Access Control) is a generalization of the RBAC model. Instead of handling roles, the ABAC model deals with attributes, in a way that subject's requests on objects are granted or denied based on assigned attributes, environment conditions, and a set of policies that are specified in terms of those attributes [Hu et al. 2014].

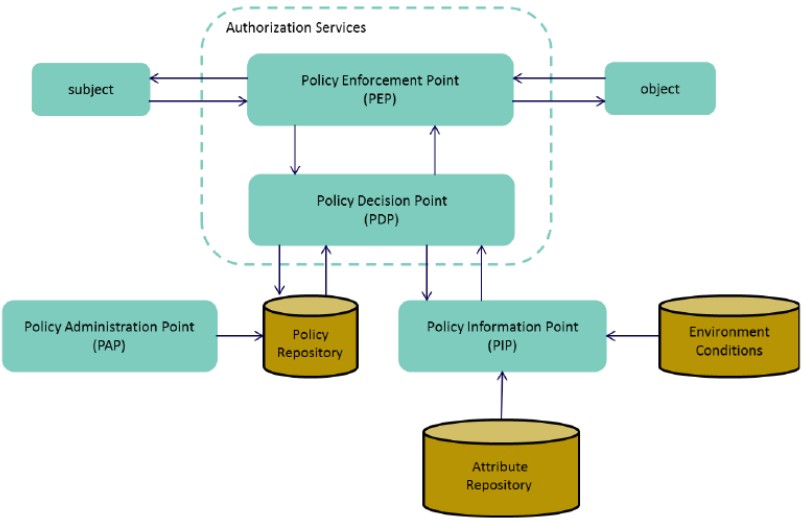


Figure 3. General view of ABAC/RBAC functional points [Hu et al. 2014].

The ABAC model defined a set of components for its access control mechanism. Figure 3 presents these components. When a subject wishes to access an object, its request goes through Policy Enforcement Point (PEP). The PEP works as a gateway, enforcing policy decisions in response to a subject request to the protected object. PEP communicates with the Policy Decision Point (PDP) in order to guarantee that the user can access the required resource. The PDP component analyses several information, including subject attributes, environment conditions and others to make a decision. For that, the PDP

queries the Policy Information Point (PIP) for subject attributes (from an attribute repository) and environment conditions, and a Policy Repository for access control policies. Based on the information collected, the PDP makes a decision that is propagated to PEP, which allows or negates access to the object. A Policy Administration Point (PAP) manages policies.

3. Overview of the Approach

The two issues presented in Section 2.2 calls for an hierarchical solution that could be applied throughout the federation, easing the management of the different services provided by RNP. Based on this, we have envisioned a solution based on an hierarchy of administrators, which can be seen in Figure 4. In our approach, we consider the existence of a Federated Access Control System (FACS) maintained by RNP as the manager of the federation. The FACS is used by SP and IdP administrators to manage access to different services.

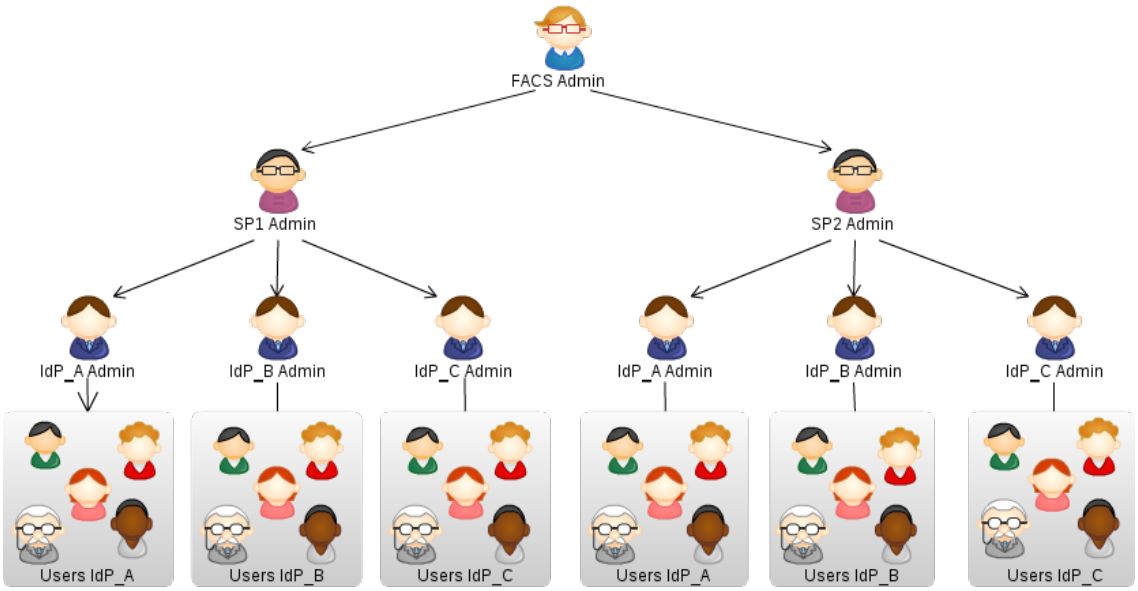


Figure 4. Hierarchical management view.

The FACS administrator is responsible for registering both SPs and IdPs, together with their respective administrators. Each SP administrator can grant IdPs access to its services considering the different access scenarios (i.e., second and third scenarios of Section 2.2). The SP administrator can also delegate access decisions to IdP administrators, which manages its IdP users access to the service, with or without an associated restriction (e.g., number of users of the IdP that can access the service).

Besides dealing with which IdP users will have access to the service, this hierarchy is also explored for the definition of access control policies regarding different access levels.

These two issues have been considered as two levels of concerns for our solution. The first level, which we refer to as SP independent, consider the management of which IdP will have access to the SP. Its independence is related to the fact that access control policies for this situation does not depend on the specific actions an user may perform

in the service. On the other hand, the SP dependent level deals with the different access control policies in place for the SP, whom which one must know details about the service in order to define a policy.

The FACS architecture, presented in Figure 5, can be seen according to this division. It is important to mention that we employ the NIST ABAC definitions [Hu et al. 2014], described in Section 2.3, when presenting some components and their respective roles. FACS is composed by several modules that will be presented in the following. The *System Administration module* is the module used by the *FACS Admin*, which represents the system administrator. This module is used to register the IdPs and SPs of the federation, as well as their respective administrators. The *IdP Management Module* is the module used by the SP administrator (*SP Admin*) to manage (e.g., add, remove, update) information about his trusted IdPs. The *Access Management Module* is the module used by both, SP and IdP Administrators, for managing user access to SPs, and privileges within SPs. The *PIP* (Policy Information Point) represents the different IdPs of the federation.

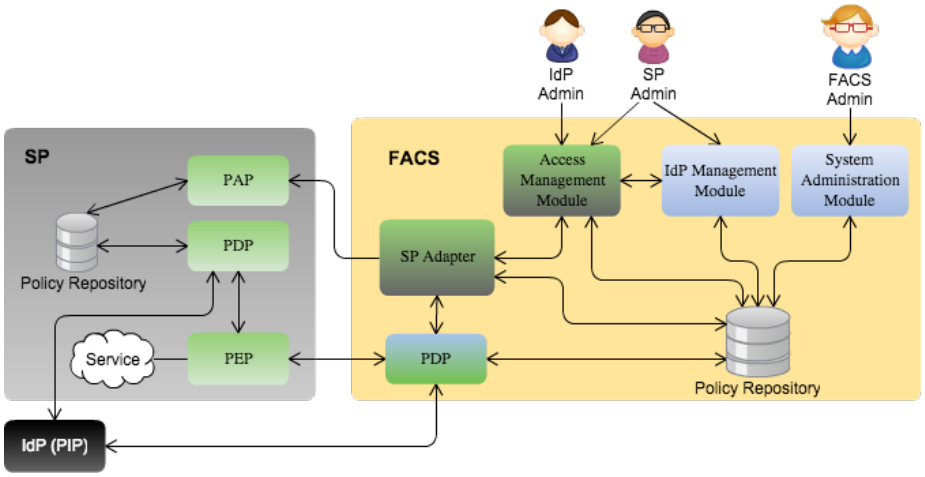


Figure 5. General architecture of FACS.

FACS deals with SP independent and SP dependent policies. SP independent policies are stored internally in FACS, and used when making decisions about an user access to the SP. SP dependent policies are propagated to the SP by means of a *SP Adapter*. This component is specific to the SP, as it must be able to communicate with the SP's PAP (Policy Administration Point) in order to update its policies, understand the policy language employed by the SP, and their different concepts, which are used to define the policy rules.

On the side of the SP, the PEP (Policy Enforcement Point) module is responsible for enforcing requests related to the SP users (i.e., users of the federation). In our approach, this component needs to be customized to communicate with FACS PDP in order to query whether or not the user has access to the service. As it happens with other SPs, it is necessary to maintain a local base, which is then used for access control management and for helping the provision of the service. Among the information kept in this base, we have quota policies, files ownership, and file sharing information. This base is represented by the *Policy Repository* in the SP.

In the sequence, we present in Figure 6 the workflow of FACS. Once an user is authenticated by its IdP, the SP PEP enters in action (activity 1), capturing the attributes emitted by the user's IdP to enforce a decision. As we said before, this decision is related to whether the user has access to the service being offered. If the user has access to the service, which can be identified by querying the SP PDP, the PEP grants access (activity 2). In case the user has no access to the service, the SP PEP needs to communicate with the PDP in FACS. At this step two checks are made. First is whether the user's IdP is registered to access the SP (activity 3). If the answer is no, then the user will be denied access to the system (activity 4). In case of yes, a second check is made (activity 5). The second check concerns the different ways an user, coming from a particular IdP, may have its access allowed.

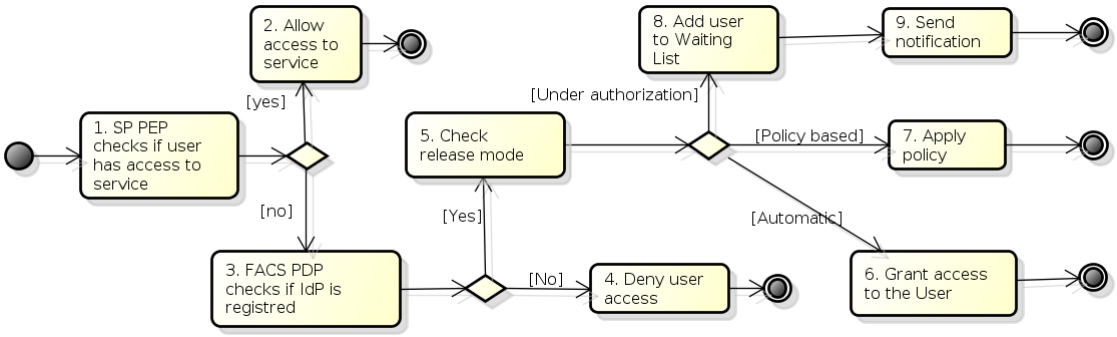


Figure 6. Workflow for access decision with FACS.

We envision an spectrum that ranges from automatic to fully manual access. In one end of this spectrum, the automatic mode, we deal with the situation where all users coming from one IdP have access to SP resources (activity 6). In an intermediary mode, access decisions could be made by means of policies, e.g., ABAC policies (activity 7). Although the decision in this mode can be made in an automatic manner, the policies still need to be defined by SP or IdP administrators. On the other hand, a fully manual mode would involve the addition of users into a waiting list (activity 8). This list would then be evaluated by SP or IdP admins, which decides on an user-by-user basis. Once a decision is made, FACS sends the user a notification (activity 9).

Anyhow, once an user has its access authorized, the SP Adapter is activated to update the user information in the SP Policy Repository. From this point on, access decisions related to the SP user, concerning both access to the service, as well as its privileges within the service, are made by the SP PDP.

4. Case Study

This section presents the implementation details of FACS, as well as its application for managing access control to users of CNC cloud, followed by with a discussion about the results obtained.

4.1. Implementation Details

The SE-CNC cloud storage service is based on the ownCloud software, an open-source software for storing files in different back-end infrastructures, supporting functionalities

like file sharing, and client synchronization. It supports several back-ends, such as, ftp, local files system, WebDav, and Openstack Swift, which is the case of SE-CNC cloud.

ownCloud is developed in php and java script languages, employs and a SQL database, and has 3 different clients: Web-based, desktop (Windows, Mac and Linux) and Mobile clients (IOS and Android). The desktop client provides synchronization capabilities. In fact, ownCloud is a complete development framework, with its internal architecture divided in several *apps*. There are several apps that can be attached to an ownCloud installation, such as apps for adding calendar and task management capabilities. It is also possible to develop an app in case a particular functionality is not provided, or even, to alter internal ownCloud workings.

Figure 7 presents a mapping of the components proposed by FACS (presented in Figure 5) and the modules of ownCloud. We consider ownCloud to be our SP domain, abstracting away from the underlying Openstack cloud infrastructure. Both ownCloud and FACS are part of the CAFe Expresso federation[4], an experimental federation provided by RNP for the development and testing of solutions.

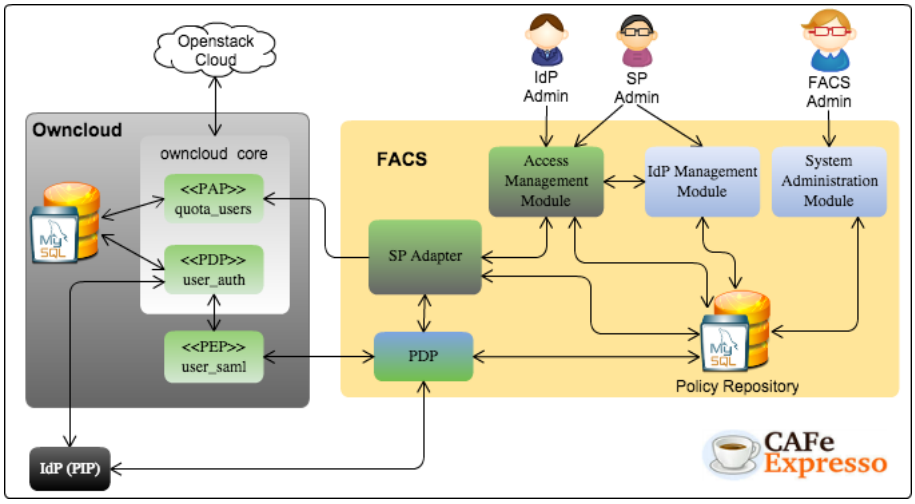


Figure 7. FACS applied to the SE-CNC cloud service.

Regarding the policy repository, ownCloud stores its data using a MySql database. The PEP module in ownCloud environment is implemented by a component called *user_saml*. This module aims to integrate our web client with federated environments, providing SAML authentication based on simpleSAMLphp software [5]. This software handle SAML assertions, the protocol employed by the CAFe Expresso federation. The PDP component is implemented by ownCloud module *user_auth*. This module is responsible for authentication and to approve user access to the service. The PAP module is implemented by ownCloud through *quota_users* module, which deals with all issues related to managing users, policies and quotas. User_auth and quota_users are part of the core owncloud module, as showed in Figure 7.

FACS is a web system implemented in php/Java script languages and its components communicates with a database, in our case, a MySql database. The three main

[4] https://wiki.rnp.br/display/gidlab/
[5] https://simplesamlphp.org/

modules of FACS have been implemented as different web interfaces, according with the role of the user accessing the system.

The *System Administration Module* can be seen as simple CRUD application, allowing the management of SPs and IdPs, and their respective administrators who will then be able to use FACS. The *IdP Management Module* supports the association of IdPs and IdP administrators to SPs, implementing the hierarchical delegation capabilities of FACS, as well as the different ways an user access is decided (automatic access, policy based or waiting list).

The *Access Management Module* is used by SPs and IdPs administrators to grant access and associate quotas to users, or groups of users. This module implementation is divided into SP independent and SP dependent. The SP independent part deals with access definition for IdP users, while its dependent part is based on the quota_users module of ownCloud. This module also supports the establishment of quotas to a particular institution, associating it to a particular IdP, which is then distributed by the IdP administrator among its users. SP Adapter is a module that knows how to interact with the SP. It also propagates the result of access control decisions.

4.2. Discussion

Different versions of FACS have been used by SE-CNC cloud service administrators for about four months, and has now reached a maturity level where it can successfully manage access control of all cloud users (a total of approximately 150 users) belonging to 14 IdPs (including the 10 IdPs from the institutions involved in the tests conducted during the experimental phase, and others that have been added for different reasons.)

Our prototype has been extensively explored considering the third scenario, in which user access is released by both SP and IdP administrators through waiting list and automatic access to particular IdPs.

Although no formal satisfaction survey has been conducted, to this moment, FACS use has presented very satisfying results. It is important to mention that the results obtained have been considered satisfactory when compared with how the access control management was conducted before FACS, and how the problems that used to arise have been mitigated. This was an error-prone process conducted by CNC administrators. Each IdP administrator would obtain its users' attributes, in collaboration with its respective users. The IdP administrator then would inform these attributes to CNC administrators, which would in turn create a script to be run against the CNC system, allowing these users to access the system.

In this context, FACS has provided an automated environment, which helped to mitigate the human errors of the process, as well as reduced the time needed to give an user access to the service. FACS has considerably simplified the effort associated with user management where, by means of its delegation capabilities, each IdP administrator is now able to manage its users without intervention from SE-CNC administrators.

5. Related Work

There are some papers that also propose approaches to handle access control in federated environments. In [Silva et al. 2014], the goal is to simplify the integration of federated

environments with distributed systems. For this, the authors have proposed an approach for access control based on scores, where the set of attributes emitted by an IdP is used to calculate a numerical value. Users are classified according with the score achieved by their attributes, which is then used to define policies to control the access to different SPs.

The SFERA project [Galheigo and Gomes 2014] considers a scenario with non-web SPs. Their approach consist of a tool for allowing federated remote access to console terminals. Similar to our approach, the IdP does not need to be modified. However, they adopt a overly simplified solution for access control, where each authenticated user is associated with a key that is deleted upon the ending of the session.

In [Ryba et al. 2013], is proposed an access control as a service applied to Smart Grids that implements SAML or XACML. An application based on this model was integrated with the Google App Engine (GAE). Due to works on a PaaS cloud, the solution interacts with applications that implement the database supported by the GAE platform. There is a work in the same field that uses cryptographic techniques to control the access and to establish security access, it uses data re-encrypt [Zhang and Chen 2012]. It solution is applied to Public cloud storage. We noticed that both approaches is only applied to a such SP or category os SPs, in contrast with FACs that is independent of SP.

FACS has some differences when compared to the solutions cited above. In general, our solution aims to supports different flavours of SPs. Differently from [Galheigo and Gomes 2014], our approach deals with privilege management. Compared to [Silva et al. 2014] solution, FACS supports different levels of administration and give flexibility to the service to establish its own policies by means of delegation.

Some of the scenarios considered for the SE-CNC service are typical of virtual organizations (VO) [Cummings et al. 2008], in the sense that only part of the users of a federation have access to the service. However, we understand that a VO presents a broader scope than the one currently being considered for FACS. Services offered by VOs can rely on authentication provided by the user's home institution, but they usually involves attributes obtained from multiple sources besides the user IdP. This multi attribute sources model requires some complex mechanism, such as support for attribute aggregation, currently no supported by FACS.

6. Conclusions and Future Work

This paper has presented an approach for managing access control to service providers in federated environments. The FACS (Federated Access Control System) provides a hierarchical privilege management solution, where access control policies definitions are delegated to IdP administrators, which can then define whose users will have access to the service.

In order to demonstrate the feasibility of our approach, we have implemented a prototype in the context of an experimental service offered by RNP. The SE-CNC cloud is a cloud-based storage service deployed in the RNP infrastructure, which requires different levels of access control. In fact, the development of our solution has been motivated by issues identified during development of SE-CNC cloud during its experimental phase, particularly, regarding its provision as a service of an academic federation. The service has been developed using the CAFe Expresso experimental federation, and has now been

migrated to the Chimarrão federation (the federation used by RNP for testing new services before they join the main CAFe federation).

Before FACS, all activities related to user access control were performed manually by the SE-CNC cloud administrators, which quickly became a burden given the crescent number of users the service was attracting. Since the deployment of FACS, the activities related to user management has been considerably simplified, mainly because of the delegation capabilities provided by FACS. An hierarchical based solution allowed us an increased flexibility, and has demonstrated to be an ideal solution for a federated environment. FACS has been able to deal with the three scenarios presented in this paper: (i) access to all users of the federation, (ii) access to particular IdPs, and (iii) access to specific users.

On the other hand, FACS has some limitations. Its use requires an implementation effort for each SP it is applied to. Although the SP dependent modules of FACS have been clearly identified and isolated, its customization process needs to be systematized and evaluated. In its current implementation, the policy based decision making for user access has been simplified to a simple attribute check. This decision was based on the scope and needs of the project at time, and now needs to be revisited. Another limitation is related to the removal of user access. Currently, FACS removes the user from its internal repository and does not propagate this change to the SP. In this way, user removal still needs to be done using the SP mechanisms. We intent to improve the SP adapter mechanism to propagate this, and eventually other, changes to the SP.

As future work, we intend to solve the limitations cited above, especially with respect to deletion of users and its propagation to the SP. In respect to policies, we envision the inclusion of a XACML policy engine, allowing an IdP administrator to specify access control policies regarding user access to services. Another goal is to apply FACS to other SPs, and to measure the effort needed to do so. We envision FACS as a template based system, which could greatly simplify the inclusion of SPs in an identity federation like the CAFE. Furthermore, we intend to extend FACS to be more generic, supporting scenarios such as Virtual Organizations (VOs), looking into new requirements such as mechanisms for attribute aggregation, currently not supported.

References

Bhargav-Spantzel, A., Camenisch, J., Gross, T., and Sommer, D. (2007). User Centricity: A Taxonomy and Open Issues. *J. Comput. Secur.*, 15(5):493–527.

Chadwick, D. W. (2009). Federated Identity Management. In *Foundations of Security Analysis and Design V*, volume 5705 of *Lecture Notes in Computer Science*, pages 96–120. Springer Berlin Heidelberg.

Cummings, J., Finholt, T., Foster, I., Kesselman, C., and Lawrence, K. (2008). Beyond Being There: A Blueprint for Advancing the Design, Development, and Evaluation of Virtual Organizations. Final report from the NSF workshops on Building Effective Virtual Organizations. Technical report, National Science Foundation.

Galheigo, M. M. and Gomes, A. T. A. (2014). Um Estudo Sobre Autenticação Federada no Acesso a Recursos Computacionais por Terminal Remoto Seguro. In *XIV Simposio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg 2014*.

Hu, V. C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., and Scarfone, K. (2014). SP 800-162. Guide to Attribute Based Access Control (ABAC) Definitions and Considerations. Technical report, National Institute of Standards and Technology, McLean and Clifton, VA, United States.

Jøsang, A., Fabre, J., Hay, B., Dalziel, J., and Pope, S. (2005). Trust Requirements in Identity Management. In *Proceedings of the 2005 Australasian Workshop on Grid Computing and e-Research - Volume 44*, ACSW Frontiers '05, pages 99–108, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Jøsang, A. and Pope, S. (2005). User Centric Identity Management. In *AusCERT Asia Pacific Information Technology Security Conference*, page 77.

Mell, P. M. and Grance, T. (2011). SP 800-145. The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, United States.

Moreira, E. Q. a., Éverton Didoné Foscarini, da Silva Junior, G. C., Alixandrina, L. A. O., Neto, L. P. V., and Rossetto, S. (2014). *Federação CAFe implantação do provedor de identidade*. RNP/ESR.

Ryba, G., Jung, M., and Kastner, W. (2013). Authorization as a service in smart grids: Evaluating the PaaS paradigm for XACML policy decision points. In *Emerging Technologies Factory Automation (ETFA), 2013 IEEE 18th Conference on*, pages 1–4.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-Based Access Control Models. *Computer*, 29(2):38–47.

Silva, E. F., Muchaluat-Saade, D., and Fernandes, N. C. (2014). Controle de Acesso Baseado em Políticas e Atributos para Federações de Recusos. In *XIV Simposio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg 2014*.

Zhang, Y. and Chen, J.-L. (2012). Access Control as a Service for Public Cloud Storage. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 526–536.

Uma Estratégia de Encaminhamento de Pacotes Baseada em Aprendizado por Reforço para Redes Orientadas a Conteúdo

Ian V. Bastos[1], **Victor C. M. Sousa**[1] **e Igor M. Moraes**[1]

[1]Laboratório MídiaCom, PGC-TCC
Instituto de Computação - Universidade Federal Fluminense
Niterói, Rio de Janeiro, Brasil

ianvilar@id.uff.br, vcmsousa@id.uff.br ,igor@ic.uff.br

Abstract. *This paper proposes a packet forwarding strategy for ICN. Our proposal is based on reinforcement learning techniques and aims at balancing the exploration of new paths and data acquired from previous exploitations. The output interfaces of a node are classified according to the content retrieval time and all interests that share the same prefix with contents previously forwarded are sent through the interface with the lowest mean retrieval time. The path exploration is probabilistic. Each node sends the same interest through the best interface and through another interface chosen at random simultaneously. The goal is to retrieve the content by using the best path found until present moment and at the same time explore copies that are recently stored in the cache of nearest nodes. Simulation results shows that the proposed strategy reduces up to 28% the number of hops traversed by received contents and up to 80% the interest load per node in comparison to other forwarding strategies.*

Resumo. *Esse artigo propõe uma estratégia de encaminhamento de pacotes para redes orientadas a conteúdo. A proposta usa a técnica de aprendizado por reforço, cuja ideia principal é realizar um balanço entre explorar novos caminhos e se aproveitar da informação adquirida durante explorações anteriores. As interfaces são classificadas com base no tempo de recuperação dos conteúdos e todo interesse com o mesmo prefixo para um conteúdo já encaminhado é enviado pela interface com o menor tempo de recuperação médio. A exploração é realizada probabilisticamente, na qual cada nó envia o mesmo interesse para a interface melhor classificada e também para uma outra interface escolhida aleatoriamente. O objetivo é fazer com que o conteúdo seja entregue pelo melhor caminho encontrado até o momento e ao mesmo tempo explorar cópias que possam ter sido armazenadas em caches ainda mais próximos recentemente. Os resultados de simulação mostram que a estratégia proposta reduz o número de saltos cerca de 28% em cada nó e 80% a carga de interesses também por nó em determinados cenários quando comparada às outras estratégias de encaminhamento.*

1. Introdução

Existe um conflito conceitual entre a rápida expansão da distribuição de conteúdos e a arquitetura orientada à conexão da Internet atual. Os usuários atualmente estão mais interessados no conteúdo em si do que na localização ou na identificação de quem o envia.

Aplicações de distribuição de conteúdo já representam mais de 60% do tráfego atual da Internet [Sandvine 2014]. Para aumentar a eficiência da distribuição de conteúdo, muitas arquiteturas baseadas no paradigma das redes orientadas a conteúdo (ROCs) foram propostas [de Brito et al. 2012]. As ROCs introduzem um novo paradigma de comunicação para redes de computadores, no qual a comunicação passa a ser centrada nos conteúdos de interesse, em vez de endereços de destino. Dessa forma, todo conteúdo que faz parte do tráfego nesse novo paradigma é identificado, endereçado e recuperado através de seu nome ao invés de sua localização física. Todos os nós da rede potencialmente armazenam o conteúdo que encaminham, com o objetivo de servir futuras requisições para o mesmo conteúdo, criando uma rede distribuída de *caches*.

Uma arquitetura proposta para as ROCs é a *Content-Centric Networking* (CCN) [Jacobson et al. 2009]. Na CCN, o processo de entrega de conteúdos é iniciado através do envio de requisições pelos usuários interessados em um dado conteúdo. Essas requisições são encaminhadas em direção a uma cópia do conteúdo requisitado, que pode estar armazenada em qualquer nó da rede que previamente tenha encaminhado esse conteúdo. Como um nó pode encaminhar uma solicitação de conteúdo para um ou mais nós vizinhos, a estratégia mais eficiente para recuperar o conteúdo é enviar a solicitação para todos os vizinhos. Caso não existisse uma limitação na largura de banda, a inundação da rede certamente garantiria a entrega da réplica com o menor atraso possível. Entretanto, os enlaces que são utilizados para propagar tais solicitações de conteúdo possuem largura de banda limitada e, em sua grande maioria, heterogênea. Em vista disso, se faz necessário o desenvolvimento de uma estratégia de encaminhamento mais inteligente do que simplesmente realizar uma inundação na rede que poderá levar a um uso ineficiente dos recursos, como a própria capacidade do enlace e espaço em *cache* [Chiocchetti et al. 2012].

Na presença de uma infraestrutura altamente distribuída de *caching* que a CCN introduz, a disponibilidade e a localização dos conteúdos são dinâmicas ao longo do tempo devido às réplicas temporárias que estão distribuídas por toda extensão da rede e a falta de coordenação entre os *caches* [Rosensweig et al. 2013]. Os fatores de distribuição e volatilidade das réplicas são determinados principalmente pela popularidade e políticas de *caching*, que por sua vez estão fortemente relacionados com a estratégia de encaminhamento adotada. Considerando os fatores mencionados, esse artigo propõe uma estratégia de encaminhamento que emprega a técnica de aprendizado por reforço, chamada de MAB (*Multi-Armed Bandit*). Cada vez que uma solicitação de conteúdo precisa ser encaminhada é preciso escolher entre n opções de interfaces de saída diferentes. Ao receber pela primeira vez a solicitação por um conteúdo nunca antes encaminhado, todas as interfaces de saídas são inundadas. Com a solicitação atendida e o conteúdo recuperado, a escolha é feita aleatoriamente entre somente a interface de saída que recuperou o conteúdo mais rapidamente, ou essa mesma interface além de uma outra selecionada ao acaso com distribuição uniforme. Após determinar a interface de saída e receber a réplica do conteúdo desejado, a estratégia proposta atribui uma recompensa numérica a essa interface, nesse caso o tempo de recuperação - tempo definido como o que leva para obter o conteúdo assim que uma requisição por ele é enviada na rede - para obter a réplica, associando a interface utilizada e o conteúdo. O objetivo é fazer com que o conteúdo seja entregue pelos nós cujo conjunto de interfaces de saída resultaram na obtenção do conteúdo com o menor tempo de recuperação com base em explorações anteriores.

Este trabalho também analisa o desempenho do mecanismo proposto através da sua implementação no simulador ndnSIM, no qual cenários de simulação foram gerados para avaliar o impacto do número de consumidores assim como a presença de falhas de enlace na rede. Para tal, métricas como carga na rede, tanto de pacotes de interesse e de dados, quantidade de saltos e o tempo de recuperação dos conteúdos foram avaliados. Os resultados mostram que nossa proposta é capaz de reduzir em cerca de 28% a quantidade saltos necessários para obter um conteúdo em cada nó da rede, assim como 75% a carga de pacotes de dados quando comparada às estratégias *Smart Flooding* e *Best Route*.

O restante do artigo está organizado da seguinte forma. Os aspectos e características básicas do funcionamento da CCN são discutidos na Seção 2. A Seção 3 introduz os conceitos estudados do problema das máquinas caça-níqueis e os contextualiza com as redes CCN e a implementação da estratégia proposta. A Seção 4 discute os parâmetros e cenários de simulação e a Seção 5 os resultados obtidos. Finalmente, na Seção 6, apresenta-se os trabalhos relacionados e a Seção 7 conclui o trabalho realizado e mostra direções para trabalhos futuros.

2. Visão Geral da Arquitetura CCN

A CCN (*Content-Centric Networking* - CCN) [Jacobson et al. 2009] é uma arquitetura de rede projetada com base no paradigma das ROCs. Em vez de endereçar conteúdos através de sua localização, a CCN os referencia pelos seus nomes legíveis. O nome de um conteúdo é composto por um ou mais prefixos de tamanho variável que são opacos para a rede. Os limites de cada prefixo são explicitamente delimitados por uma barra "/". Em um exemplo, o nome do conteúdo que referencia a página principal do Instituto de Computação da Universidade Federal Fluminense pode ser representado por "/uff.br/ic/index.html". Conteúdos muito grandes devem ser divididos em fragmentos menores denominados *chunks*. Uma requisição através do prefixo "/uff.br/ic/redes/aula1.avi/4", por exemplo, se remete ao quarto fragmento do conteúdo aula1.avi da disciplina redes de computadores ministrada no Instituto de Computação da Universidade Federal Fluminense.

Um usuário final na CCN realiza requisições na rede ao enviar pacotes de interesse (*Interest Packets*) para os conteúdos nomeados. Os pacotes são encaminhadas salto a salto em direção a uma cópia permanente do conteúdo requisitado. Para cada interesse recebido, a entidade CCN (roteador ou sistema final) realiza uma consulta por nomes de conteúdos na tabela de encaminhamento (*Forwarding Interest Base* - FIB), que armazena o conjunto de interfaces por qual um determinado conteúdo pode ser obtido. Como vários caminhos são possivelmente conhecidos para um certo prefixo, a camada de estratégia fica responsável por selecionar uma ou mais interfaces de próximo salto entre o conjunto de possibilidades. A camada de estratégia é o módulo responsável por realizar as retransmissões assim como selecionar quais e quantas das várias interfaces de saída serão utilizadas para encaminhar o interesse [Jacobson et al. 2009]. Nós intermediários no caminho escolhido podem dispor de cópias armazenadas em *cache* do conteúdo de interesse em seu armazém de conteúdos (*Content Store* - CS). Nesse caso, pacotes de interesse não necessitam ser encaminhados até o repositório que detém a cópia permanente, e a réplica temporária é enviada imediatamente pelo caminho inverso até o usuário que a requisitou utilizando as informações contidas nas tabelas de interesse pendente (*Pending Interest Table* - PIT). Essa tabela possui como responsabilidade manter uma indexação

entre os prefixos encaminhados pelo nó e todas as interfaces de entrada por onde pacotes de interesse de um mesmo conteúdo foram recebidos. Dessa forma, todo roteador ao encaminhar um pacote de interesse possui a garantia que o pacote de dados correspondente irá passar por ele no caminho de volta até o consumidor que o requisitou.

3. Estratégia de Encaminhamento Proposta

O objetivo da estratégia proposta, chamada de MAB, é minimizar o tempo de recuperação de conteúdos observado pelo usuário final. A ideia principal é descobrir caminhos alternativos que levem a réplicas temporárias de conteúdos sem que nenhuma informação sobre a localização desses conteúdos se encontre disponível a *priori*. A estratégia proposta modela o encaminhamento como o problema das Máquinas Caça-Níqueis e desenvolve um algoritmo em que, além de explorar probabilisticamente suas interfaces de saída, entrega o conteúdo de interesse pela interface que obteve o menor tempo de recuperação.

3.1. O Problema das Máquinas Caça-Níqueis e a Construção da Tabela de Encaminhamento

O problema das Máquinas Caça-Níqueis (*Multi-Armed Bandit*) é um clássico problema encontrado na teoria da probabilidade no qual um apostador, na presença de múltiplas máquinas caça-níqueis, precisa decidir em qual máquina irá apostar. Quando utilizada, cada máquina fornece uma recompensa aleatória a partir de uma distribuição estatística específica da máquina selecionada. O objetivo do apostador é maximizar seu ganho ao concentrar suas apostas nas máquinas que produzem melhores recompensas sobre um determinado período de tempo [Barto 1998].

No contexto da CCN, as máquinas podem ser associadas às interfaces presentes em cada nó da rede. Toda vez que um interesse é recebido por um nó e o conteúdo não se encontra presente em seu *cache*, e não possui uma entrada na PIT indicando um encaminhamento anterior para o mesmo interesse, o nó estará diante do mesmo problema que o apostador ao precisar escolher uma ou mais interfaces para encaminhá-lo.

Na estratégia proposta, que é baseada no algoritmo ε-*greedy*, cada nó constrói sua tabela de encaminhamento ao registrar o tempo de recuperação de cada prefixo de nome quando recebe o pacote de dados correspondente por uma das interfaces de próximo salto. A tabela é composta por entradas que indicam o valor de recompensa médio *Q* em cada nó *i* para todos os destinos *d*, $Q_i(f, d) \forall d \in interfaces(i)$, onde $Q_i(f, d)$ representa o tempo de recuperação médio para obter o conteúdo *f* após o nó *i* ter enviado um pacote de interesse via interface que alcança *d*. A ação de encaminhar realizada por um nó *i* consiste em selecionar a interface de próximo salto *d* com a menor recompensa média *Q* para um conteúdo *f*, ou seja, a interface que ao longo do tempo produziu o menor tempo de recuperação para que o conteúdo pudesse ser obtido. Descobrir e selecionar a interface de próximo salto d possui complexidade $O(|interfaces(i)|)$. Para cada pacote de dados recebido de volta pela interface *d*, o nó *i* atualiza a recompensa *Q* em sua tabela de acordo com a Equação 1, onde $rtt_{(f,d),k}$ representa o atraso desde o nó *i* encaminhar o pacote de interesse pelo conteúdo *f* até o momento em que *i* recebe o pacote de dados que contém *f*.

Cada nó atualiza sua tabela de encaminhamento independentemente e, ao serem utilizados em conjunto, conseguem identificar o "melhor" caminho dada as condições

momentâneas da rede para encaminhar e recuperar um determinado conteúdo. Para cada conteúdo $f \in F$, conjunto de todos conteúdos disponibilizados, um nó i mantém um conjunto de valores $Q_i(f, d) \forall d \in interfaces(i)$, que são computados e atualizados toda vez que o nó *i* recupera f, seja pela "melhor" interface ou por uma interface explorada, como será descrito a seguir.

3.2. O Processo de Encaminhamento e Exploração

A estratégia de encaminhamento proposta supõe que nenhum conhecimento prévio foi adquirido antes de receber o interesse por um conteúdo pela primeira vez. Posto isso, ao receber um pacote de interesse por um conteúdo f que não consta na tabela de encaminhamento, o roteador *i* irá inundar todas as suas interfaces, exceto a que recebeu o interesse, para inicializar o valor das recompensas $Q_i(f, d)$. O objetivo de tal inundação está em adquirir conhecimento sobre o estado atual da rede em relação a *f*, garantindo ao mesmo tempo a entrega dos pacotes de dados requisitados.

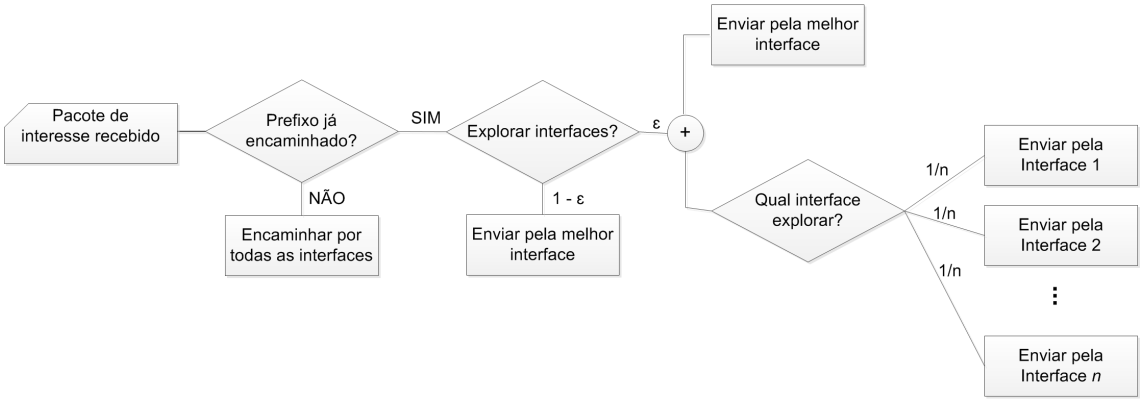


Figura 1. Processo decisório ao encaminhar um pacote de interesse.

Após essa fase inicial, as interfaces são classificadas de acordo com o tempo transcorrido para a recuperação dos primeiros pacotes de dados. A partir desse ponto, toda vez que um pacote de interesse com o prefixo de f for recebido por um nó, ele pesquisará pela interface que é considerada a melhor classificada para recuperar o conteúdo solicitado com tempo constante em sua tabela de encaminhamento através do *hash* computado a partir do prefixo. Em seguida, encaminhará o interesse por um processo aleatório, que decide com uma probabilidade $1 - \varepsilon$ propagar o pacote de interesse somente pela interface que produziu o menor valor $Q_i(f, d)$ no decorrer do tempo. Ou, com probabilidade ε, propagar o interesse para a "melhor" interface e também para outra interface escolhida ao acaso diferente da atual "melhor" seguindo uma distribuição uniforme. Através dessa abordagem, garante-se que f será entregue pelo *cache* mais "próximo" encontrado até o momento enquanto outras cópias que possam ter sido armazenadas por *caches* ainda mais "próximos" em um passado mais recente possam ser encontradas. A opção de possuir somente uma única interface para exploração foi adotada para limitar a sobrecarga de pacotes de interesse trafegando na rede [Chiocchetti et al. 2013].

Da mesma forma, para cada pacote de dados retornado passada a inundação, os nós atualizarão sua tabela de encaminhamento de acordo com a seguinte equação [Barto 1998]:

$$
\begin{aligned}
Q_{i,k+1}(f,d) &= Q_{i,k}(f,d) + \alpha[rtt_{(f,d),k} - Q_{i,k}(f,d)] \\
&= \alpha rtt_{(f,d),k} + (1-\alpha)\, Q_{i,k}(f,d) \\
&= \alpha rtt_{(f,d),k} + (1-\alpha)[\alpha rtt_{(f,d),k-1} + (1-\alpha)\, Q_{i,k-1}(f,d) \\
&= \alpha rtt_{(f,d),k} + (1-\alpha)\alpha rtt_{(f,d),k-1} + (1-\alpha)^2 Q_{i,k-1}(f,d) \\
&= \alpha rtt_{(f,d),k} + (1-\alpha)\alpha rtt_{(f,d),k-1} + (1-\alpha)^2 \alpha rtt_{(f,d),k-2} + \\
&\cdots + (1-\alpha)^{k-1}\alpha rtt_{(f,d),k-1} + (1-\alpha)^k Q_{i,1}(f,d) \\
&= (1-\alpha)^k Q_{i,1}(f,d) + \sum_{j=1}^{k} \alpha(1-\alpha)^{k-j} rtt_{(f,d),j}.
\end{aligned}
\tag{1}
$$

Como a localização das réplicas nos *caches* muda com o passar do tempo, incorpora-se uma atualização que leva em conta a não-estacionariedade do problema de encontrar o *cache* que retorna mais rapidamente o conteúdo requisitado. Fixando o valor de α em uma constante, tal que $0 < \alpha \leq 1$, a Equação 1 mostra que, como $1 - \alpha$ é menor que 1, o peso atribuído a $rtt_{(f,d),j}$ diminui à medida que a quantidade de tempos de recuperação intervenientes coletados aumentam. Na verdade, o peso decai exponencialmente de acordo com o expoente $1 - \alpha$. Consequentemente, a recuperação de um pacote de dados no estado atual da rede sempre possuirá um peso maior no cálculo do tempo de recuperação médio, mas sem que o histórico obtido a partir de uma interface deixe de ser levado em conta, já que o estado corrente pode apresentar uma condição particular e temporária.

Outro tratamento realizado via recepção de um pacote de dados é a identificação de uma possível queda de enlace. Cada entrada da tabela de encaminhamento possui uma tupla contendo o prefixo de um conteúdo, a interface utilizada na propagação do interesse, o tempo de recuperação médio e um contador de retransmissões. Assim, sempre que um interesse é enviado, sua entrada correspondente na tabela tem o contador acrescido de uma unidade. Caso esse contador atinja um valor de limiar predefinido *T*, o valor do tempo de recuperação médio é definido como infinito e, consequentemente, a interface associada a essa entrada deixa de ser utilizada. Como a estratégia de encaminhamento proposta realiza explorações de forma aleatória, eventualmente tal interface que foi classificada como "fora do ar" acaba sendo sondada novamente e, no recebimento de um respectivo pacote de dados, todo processo de aquisição de recompensas associado aos prefixos previamente encaminhados por ela recomeça.

4. Simulação

Para que o desempenho da estratégia proposta pudesse ser avaliado e comparado, o simulador ndnSIM [Afanasyev et al. 2012] foi estendido para acomodar a MAB. A estratégia proposta foi comparada com duas outras estratégias de encaminhamento implementadas por padrão no ndnSIM: a *Best Route* e a *Smart Flooding*. Ambas estratégias apoiam-se em uma codificação de cor para cada interface de saída ao produzir estatísticas das informações coletadas no plano de dados, como tempo de recuperação e sobrecarga da PIT. A primeira encaminha interesses somente para a interface de cor verde (teve o interesse enviado e o dado correspondente retornado) melhor classificada se disponível. Caso

contrário, uma das interfaces de cor amarela (estado da interface desconhecido, ainda não utilizada) é selecionada. A segunda segue o mesmo princípio, entretanto no caso de não possuir uma interface de cor verde, inunda todas as interface de cor amarela. As seguintes métricas são utilizadas para avaliar o desempenho das estratégias de encaminhamento:

- **número médio de saltos:** número médio de saltos que foram necessários para obter o *chunk* de interesse para cada consumidor, incluindo o caminho percorrido pelos pacotes de interesse e pelos pacotes de dados.
- **tempo de recuperação médio:** tempo decorrido, em milissegundos, a partir do primeiro interesse enviado até que o *chunk* requisitado fosse entregue com sucesso.
- **número médio de pacotes de interesse:** média da carga de interesses inseridos na rede.
- **número médio de pacotes de dados:** média da carga de dados inseridos na rede.

A topologia de rede usada na avaliação é uma topologia real obtida através do mapeador de topologias *Rocketfuel* [Spring et al. 2002]. Essa topologia possui um total de 163 nós, dos quais 72 pertencem a borda da rede. A largura de banda dos enlaces, assim como o tamanho da fila das interfaces e o atraso de transmissão são heterogêneos. Para efeito de comparação com as estratégias já adotadas pelo simulador, os experimentos são divididos em função do número de consumidores. Além disso, testes são realizados em diferentes configurações de consumidores para avaliar o comportamento das estratégias de encaminhamento na presença de falha de enlaces.

Em todos os experimentos, existem 3 produtores fixos e, com exceção da avaliação do número de consumidores, 20 consumidores selecionados aleatoriamente entre os 72 nós de borda. Cada consumidor envia 100 interesses por segundo seguindo a distribuição de popularidade MandelBrot-Zipf [Silagadze 1999] com α = 0,7. O *cache* dos nós tem capacidade para armazenar até 1000 *chunks* e cada *chunk* tem tamanho igual a 1024 KB. A *Least Recently Used* (LRU) é a política de descarte empregada pelos *caches*. O limiar de retransmissões T para identificação de um possível problema na rede é configurado em 50 tentativas para a estratégia de encaminhamento proposta. Os demais parâmetros do simulador foram mantidos em sua configuração padrão. Todas as simulações tiveram uma duração de 60 s e os resultados são gerados através da média de mútliplas rodadas de simulação, com os intervalos de confiança calculados, representados por barras verticais, para um nível de confiabilidade de 95%.

5. Resultados

5.1. Definição dos Parâmetros da Estratégia de Encaminhamento Proposta

O primeiro passo é investigar o impacto dos parâmetros da estratégia de encaminhamento proposta no seu desempenho. Primeiramente varia-se o parâmetro α para verificar o peso adequado de importância a ser colocado no histórico de recompensas (tempo de recuperação) coletados ao longo do tempo. Nesse experimento, para efeito de simplificação, ajustamos o parâmetro ε em 0,5, ou seja, é concedida a mesma chance de utilizar somente o melhor caminho e de explorar caminhos alternativos. Os resultados foram derivados de 20 rodadas de simulação e representam a média de cada nó consumidor.

Pode-se observar na Figura 2 que o tempo de recuperação médio, diminui conforme aumenta-se o valor de α a partir de 0,2. Esse comportamento indica que atribuir

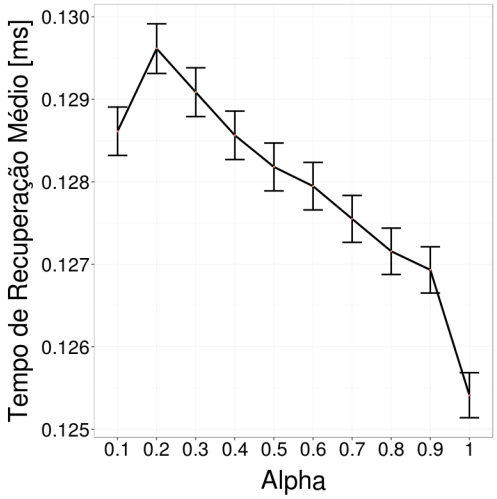


Figura 2. Atraso médio por nó em função de α.

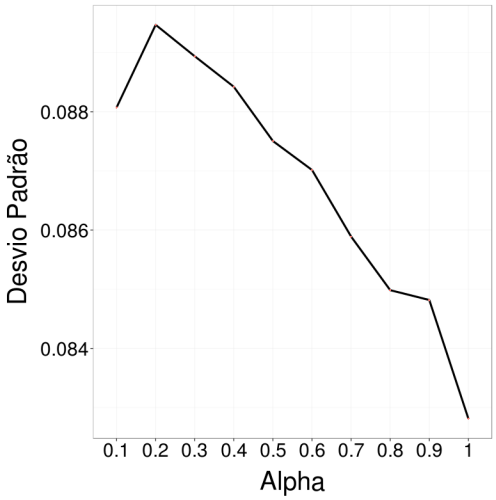


Figura 3. Desvio Padrão.

uma importância maior ao caminho que recuperou um conteúdo com menor atraso mais recentemente proporciona uma adaptação mais rápida em relação à disponibilidade dos contéudos na rede. A Figura 3 mostra que, além de diminuir o tempo de recuperação médio com maiores valor de α, também mantém-se uma maior estabilidade nos caminhos selecionados. Portanto, assume-se $\alpha = 1$ na comparação da estratégia proposta com as demais estratégias.

Com $\alpha = 1$, investiga-se o melhor valor para ε. A Figura 4 mostra que o tempo de recuperação médio diminui conforme ε aumenta, porém o alto nível de exploração diminui a estabilidade dos caminhos selecionados, ilustrado na Figura 5.

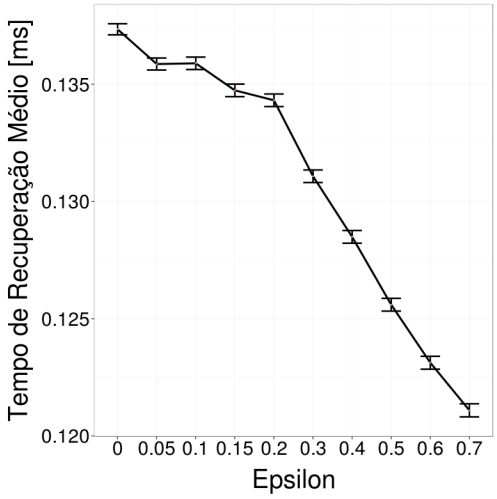


Figura 4. Atraso médio por nó em função de ε.

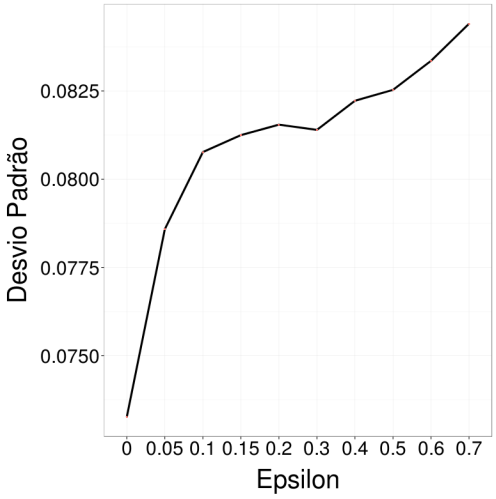


Figura 5. Desvio Padrão.

Como o tempo de recuperação médio e seu desvio padrão divergiram, analisa-se também o comportamento da quantidade de interesses satisfeitos e do número de pacotes

de interesses recebidos em média por nó da rede. A primeira métrica informa os interesses agregados na PIT de cada roteador que foram satisfeitos antes de seu temporizador estourar. A segunda informa a agregação bem sucedida de interesses na PIT. Na Figura 6, pode-se observar que a no intervalo [0,2, 0,4] a quantidade média de interesses satisfeitos aumenta e a carga média de interesses recebidos diminui. Ao se avaliar a razão entre interesses satisfeitos e interesses recebidos nesse intervalo, ε = 0,4 proporciona o maior ganho, aproximadamente 63% em relação à configuração em que nunca novos caminhos são explorados (ε = 0), e uma perda de aproximadamente 14% em relação à configuração com ε = 0,7, mas com uma menor sobrecarga.

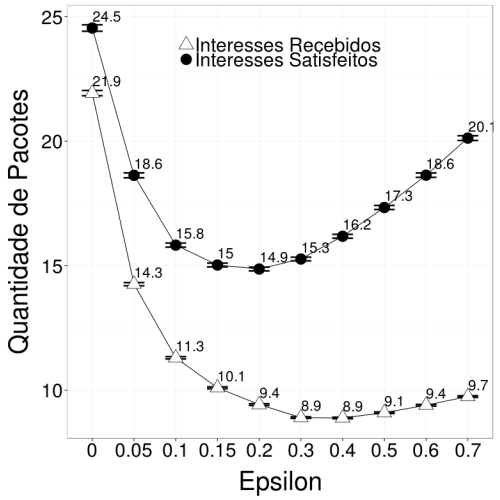


Figura 6. Média de interesses satisfeitos e interesses recebidos por nó.

5.2. Comparação das Estratégias de Encaminhamento

O objetivo dos próximos resultados é avaliar e comparar o desempenho da estratégia proposta MAB com as estratégias *Smart Flooding* (SF) e *Best Route* (BR). Avaliam-se todas as métricas descritas na Seção 4 em função do número de consumidores presentes na rede, que varia de 5 a 50 consumidores. Todos os parâmetros dos mecanismos BR e SF são mantidos do padrão e o nosso mecanismo com α = 1 e ε = 0,4.

É possível observar na Figura 7 que a MAB possui um menor tempo de recuperação médio comparada ao SF e uma curva muito semelhante ao BR. Entretanto as Figuras 8 e 9 mostram que a carga inserida na rede, tanto de pacotes de interesse e de pacotes de dados, é menor com a MAB que ambos os mecanismos. Comparada ao SF, a MAB reduz de 44-46% a quantidade de dados recebidos por nó, e com o BR de 55-75%. Já em relação aos pacotes de interesse, a MAB reduz de 21-37% comparado ao SF, desconsiderando seu pico, e cerca de 51-80% quando comparado ao BR. Esse comportamento está associado à exploração esporádica de novos caminhos realizada por cada nó da rede, fazendo com que nós vizinhos aumentem a disponibilidade dos conteúdos e, consequentemente, diminua a necessidade dos pacotes serem encaminhados por uma quantidade maior de saltos, o que pode ser verificado na Figura 10.

A segunda avaliação realizada tem como objetivo verificar como a estratégia proposta MAB se comporta na presença de falha de enlaces. Por realizar explorações esporádicas, espera-se que a MAB mantenha aproximadamente as mesmas taxas para cada

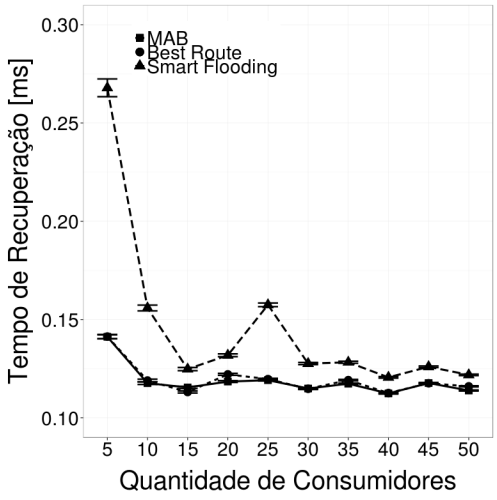


Figura 7. Atraso médio por nó em função da quantidade de consumidores.

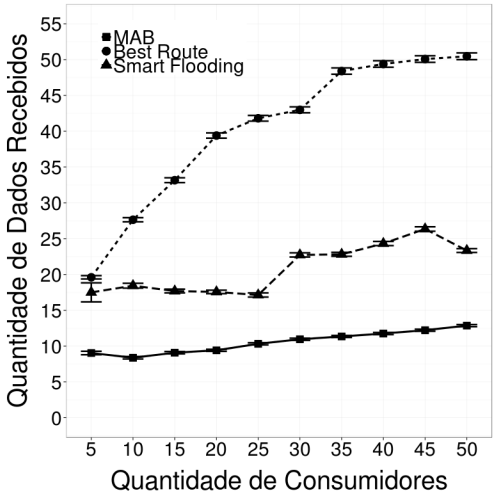


Figura 8. Média da quantidade de pacotes de dados recebidos por nó em função da quantidade de consumidores.

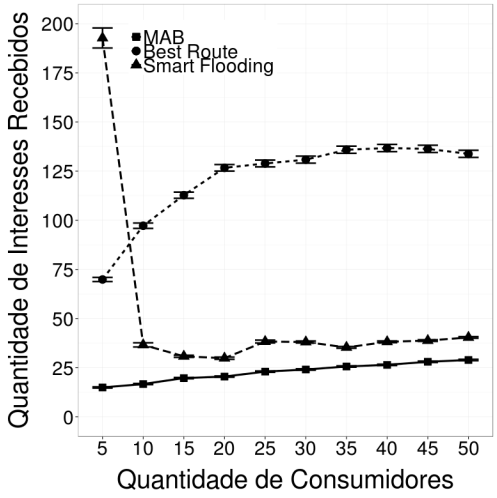


Figura 9. Média da quantidade de pacotes de interesse recebidos por nó em função da quantidade de consumidores.

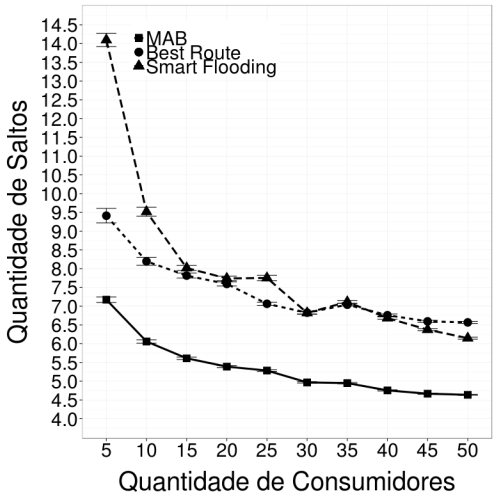


Figura 10. Média do número de saltos por nó em função da quantidade de consumidores.

métrica na presença e na ausência de pequenas falhas na rede em consequência dos caminhos alternativos encontrados durante as explorações. Para realizar a avaliação, são gerados aleatoriamente 20 cenários de falha diferentes contendo cada um 20 consumidores diferentes. Os enlaces foram quebrados em tempos escolhidos de forma aleatória e todos próximos aos produtores que se mantiveram fixos para todos os cenários, forçando a escolha de outros enlaces após as falhas.

A Figura 11 mostra que o tempo de recuperação médio com a MAB se mantém quase que idêntico aos experimentos anteriores, ficou ligeiramente maior que o do SF. Acredita-se que essa diferença seja em razão da forma como os dois mecanismos iden-

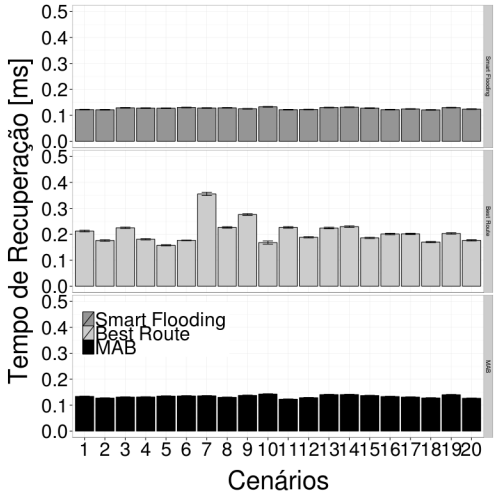


Figura 11. Atraso médio por nó para cada cenário.

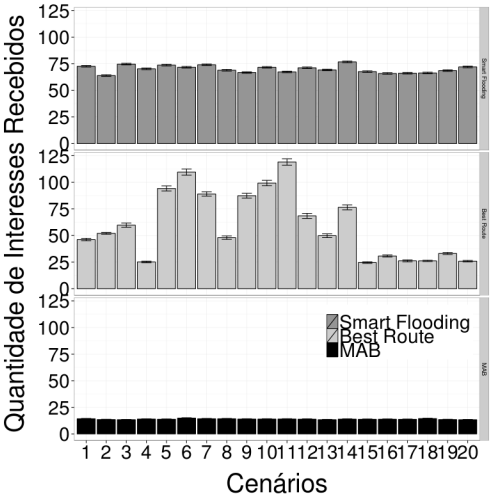


Figura 12. Média da quantidade de pacotes de interesse recebidos por nó para cada cenário.

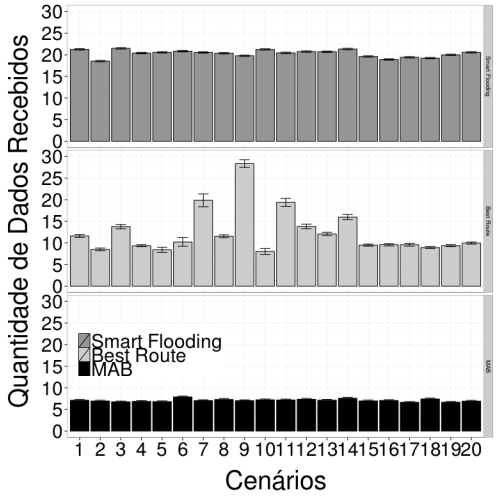


Figura 13. Média da quantidade de pacotes de dados recebidos por nó para cada cenário.

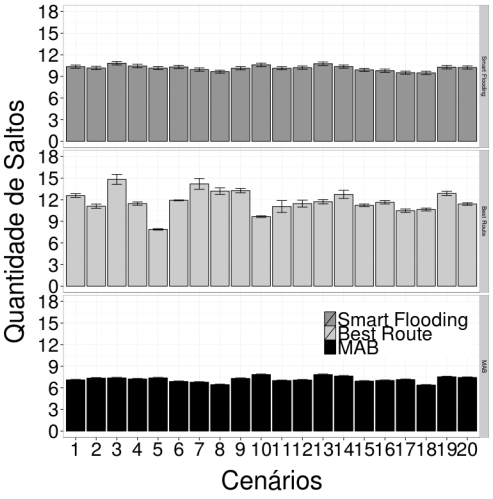


Figura 14. Média do número de saltos por nó para cada cenário.

tificam possíveis problemas na rede. O SF realiza essa identificação pelo uso de pacotes NACK, onde nós acima na hierarquia comunicam a impossibilidade de encaminhar o interesse aos nós inferiores com mais rapidez. A MAB realiza essa identificação ao perceber que depois de T retransmissões nenhum pacote de dados chegou em resposta. No caso dos experimentos, $T = 50$. Ainda assim, a MAB mantém a baixa carga média por nó de pacotes de interesse, Figura 12, e de pacotes de dados, Figura 13 comparada ao SF e ao BR. Além disso, é necessário um menor número de saltos para recuperar os conteúdos, o que mais uma vez enfatiza o aumento da disponibilidade dos conteúdos com o uso das explorações.

6. Trabalhos Relacionados

Pesquisas anteriores sobre encaminhamento em ROCs progrediram em duas direções diferentes. Por um lado protocolos de roteamento operando no plano de controle [Wang et al. 2012b, Wang et al. 2012a, Hoque et al. 2013] possuem a finalidade de disseminar as informações contidas na FIB que endereçam réplicas permanentes. O protocolo de roteamento baseado em nomes OSPFn [Wang et al. 2012a] realiza anúncio dos prefixos de nomes de conteúdos permanentes, que podem ser obtidos com os interesses encaminhados até o produtor que detém as réplicas permanentes.

Na outra direção, trabalhos com o foco em estratégias de encaminhamento no plano dos dados [Chiocchetti et al. 2013, Yi et al. 2012, Rossini e Rossi 2013] possuem como finalidade a recuperação de réplicas voláteis seguindo alguma métrica, como a mais próxima ou a com o menor tempo de recuperação. A principal vantagem dessa abordagem em relação à anterior é a possibilidade de recuperar réplicas temporárias sem a necessidade de ficar sujeito a sobrecarga de sinalização explícita para descobrir sua localização. Dessa forma, realizar uma exploração restrita na rede no intuito de ajustar o encaminhamento conforme o estado momentâneo da rede pode reduzir o tamanho das tabelas de encaminhamento [Chiocchetti et al. 2012] em troca de uma comunicação um pouco mais intensa, mas que poderá beneficiar terceiros com o aumento da disponibilidade dos conteúdo com réplicas armazenadas em nós vizinhos.

Em particular, Chiocchetti et al. 2012 evidenciam o potencial de usar somente o primeiro fragmento de um conteúdo para explorar a rede em busca de réplicas temporárias [Chiocchetti et al. 2012]. Posteriormente, partindo da ideia de explorar esporadicamente a rede, Chiocchetti et al. 2013 utiliza a técnica Q-*learning* de aprendizado por reforço distribuidamente em cada nó da rede com a troca de informações das suas recompensas ao anexá-las nos conteúdos que encaminham [Chiocchetti et al. 2013], mas não apresentam um artífício para verificar que um dado conteúdo está inacessível por alguma das interfaces de saída. A ausência de um mecanismo para identificação de falhas na rede em uma estratégia de encaminhamento que decide a melhor interface de saída com base no tempo de recuperação dos conteúdos pode prejudicar gravemente seu funcionamento. Não receber um pacote de dados implica na não atualizacão da entrada da tabela de encaminhamento correspondente a interface "defeituosa" e, no caso de ter sido classificada anteriormente como "melhor" interface, o nó continuará enviando interesses por ela sem que os conteúdos sejam recebidos. Por último, Yi et al. 2012 introduz em uma abordagem dinâmica na qual as interfaces são sondadas periodicamente e estatísticas são coletadas para cada uma delas [Yi et al. 2012]. Caso, para um dado conteúdo, uma interface se mostre "melhor" do que a utilizada atualmente, a estratégia de encaminhamento substitui para essa interface a ser explorada, entretanto, por somente utilizarem esse artífício em momentos de uma possível degradação da interface utilizada, deixam de investigar potenciais melhores caminhos.

A estratégia proposta realiza explorações probabilisticamente ao invés de usar a fase de exploração e esperar uma certa convergência como em Chiocchetti et al. 2013, mas aproveita a ideia dos autores de utilizar uma única interface de saída quando não está explorando e duas interfaces de saída quando está. Também realiza detecção de falhas na rede ao usar um limiar de retransmissões, diferentemente de Yi et al. 2012 que usam um pacote de controle.

7. Conclusão

Esse trabalho teve por objetivo avaliar uma estratégia de encaminhamento para a arquitetura CCN baseada na técnica de aprendizado por reforço $\varepsilon - greedy$. Assim sendo, foi desenvolvida a estratégia denominada MAB (*Multi-Armed Bandits*). Um nó CCN incorporado com a estratégia proposta deve primeiro classificar suas interfaces de acordo com o tempo de recuperação de um dado conteúdo. Em seguida, para cada interesse recebido com um prefixo de nome já presente na tabela de encaminhamento, o nó decide probabilisticamente se envia o interesse somente pela melhor interface classificada ou se envia também o interesse por outra interface escolhida aleatoriamente. O objetivo é aprender outras interface que levem a cópias de conteúdos em nós cada vez mais próximos e, assim, reduzir o tempo de recuperação de conteúdos.

Além do desenvolvimento da estratégia, foi realizada a comparação do seu desempenho com duas outras estratégias de encaminhamento: *Best Route* e *Smart Flooding*. A avaliação considera uma topologia de rede real chamada *Rocketfuel* e duas abordagens para análise das métricas, o número de consumidores e cenários de falha. Os resultados mostram que a estratégia proposta é a que proporciona o menor tempo de recuperação de conteúdos, pois reduz em até 28% o número médio de saltos atravessados pelos conteúdos recuperados com sucesso. Também conclui-se que a estratégia proposta é robusta contra falhas de enlace, uma vez que não aumenta significativamente o tempo de recuperação de conteúdos e a sobrecarga de controle para detectar uma falha.

Algumas avaliações não foram realizadas mas são bastante interessantes de serem feitas. É desejado investigar futuramente a influência do tamanho relativo do *cache* dado um catálogo de conteúdos fixo, assim como o efeito causado pelo tamanho da PIT na taxa de entrega de conteúdos. Além disso, deseja-se estudar como os parâmetros presentes na estratégia proposta influenciam em achar o nó mais "próximo" para conteúdos com diferentes níveis de popularidade. É interessante também analisar os prós e contras da abordagem de usar um limiar de retransmissões com a abordagem de um pacote de controle NACK para detecção de falhas.

Outras considerações importantes a serem estudadas estão relacionadas com a modelagem da simulação. Prentende-se pesquisar modelos que representem uma forma mais consistente com a realidade da Internet, como a taxa de geração de requisições por conteúdos, a distribuição da popularidade dos conteúdos e falhas de enlaces e equipamentos para que a estratégia proposta possa ser comparada com outras estratégias presentes na literatura.

Agradecimentos

Este trabalho é apoiado por CNPq, CAPES, FAPERJ, Proppi/UFF, TBE/ANEEL e CELESC/ANEEL.

Referências

Afanasyev, A., Moiseenko, I. e Zhang, L. (2012). ndnSIM: NDN simulator for NS-3. Relatório técnico, University of California, Los Angeles.

Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT press.

Chiocchetti, R., Perino, D., Carofiglio, G., Rossi, D. e Rossini, G. (2013). INFORM: a dynamic interest forwarding mechanism for information centric networking. Em *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, páginas 9–14. ACM.

Chiocchetti, R., Rossi, D., Rossini, G., Carofiglio, G. e Perino, D. (2012). Exploit the known or explore the unknown?: hamlet-like doubts in ICN. Em *Proceedings of the second edition of the ICN workshop on Information-centric networking*, páginas 7–12. ACM.

de Brito, G. M., Velloso, P. B. e 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.

Hoque, A., Amin, S. O., Alyyan, A., Zhang, B., Zhang, L. e Wang, L. (2013). NLSR: Named-data link state routing protocol. Em *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, páginas 15–20. ACM.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H. e Braynard, R. L. (2009). Networking named content. Em *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, páginas 1–12. ACM.

Rosensweig, E. J., Menasché, D. S. e Kurose, J. (2013). On the steady-state of cache networks. Em *INFOCOM, 2013 Proceedings IEEE*, páginas 863–871. IEEE.

Rossini, G. e Rossi, D. (2013). Evaluating CCN multi-path interest forwarding strategies. *Computer Communications*, 36(7):771–778.

Sandvine (2014). Global Internet phenomena report. Relatório técnico, Sandvine Incorporated ULC.

Silagadze, Z. K. (1999). Citations and the Zipf-Mandelbrot's law. Relatório técnico, arXiv preprint physics/9901035.

Spring, N., Mahajan, R. e Wetherall, D. (2002). Measuring ISP topologies with Rocketfuel. Em *ACM SIGCOMM Computer Communication Review*, volume 32, páginas 133–145. ACM.

Wang, L., Hoque, A., Yi, C., Alyyan, A. e Zhang, B. (2012a). OSPFN: An OSPF based routing protocol for named data networking. Relatório técnico, University of Memphis and University of Arizona.

Wang, Y., Lee, K., Venkataraman, B., Shamanna, R. L., Rhee, I. e Yang, S. (2012b). Advertising cached contents in the control plane: Necessity and feasibility. Em *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, páginas 286–291. IEEE.

Yi, C., Afanasyev, A., Wang, L., Zhang, B. e Zhang, L. (2012). Adaptive forwarding in named data networking. *ACM SIGCOMM computer communication review*, 42(3):62–67.

Eleição Assíncrona de Líder em Memória Compartilhada Dinâmica

Cátia Khouri[1,2], **Fabíola Greve**[2]

[1]DCET – U. Estadual do Sudoeste da Bahia (UESB), Vitória da Conquista, BA – Brasil

[2]D. de Ciência da Computação – U. Federal da Bahia (UFBA), Salvador, BA – Brasil

Abstract. *Cloud systems, SANs (storage area networks), etc. are inherently dynamic. The eventual leader detector is a powerful abstraction to ensure consistency and fault-tolerance in face of such dynamics. This paper presents a leader service, of class Ω, for the shared dynamic memory model, subject to process crashes. Differently from known solutions, it does not require the knowledge of the cardinality of the system membership, in such a way that nodes can leave and join the system as they wish.*

Resumo. *Sistemas distribuídos em nuvens, redes de área de armazenamento (SAN's), etc. são inerentemente dinâmicos. O detector de líder é uma abstração poderosa para garantir a consistência e tolerância a falhas em face de tal dinamismo. Este artigo apresenta um serviço de líder, de classe Ω, para o modelo de memória dinâmica compartilhada, sujeito a falhas de processos. Diferentemente das soluções conhecidas, ele não requer o conhecimento da cardinalidade dos participantes do sistema, de tal forma que nós podem sair e entrar no sistema livremente.*

1. Introdução

Sistemas distribuídos atuais desenvolvidos sobre redes de sensores, redes móveis *ad hoc*, grades oportunistas, computação em nuvem, redes de área de armazenamento (SAN's), redes P2P não-estruturadas, etc. possibilitam a seus participantes acessarem serviços e informações independentemente de sua localização ou mobilidade. Isso é possível através de um desenho de sistema altamente dinâmico que dispensa a existência de uma autoridade administrativa centralizada ou uma infraestrutura estática. Questões relacionadas ao desenho de soluções confiáveis que lidam com a natureza de alto dinamismo e auto-organização de tais redes são um campo muito ativo de pesquisa.

Por outro lado, no projeto de sistemas distribuídos confiáveis, camadas superiores de serviço (consenso, difusão atômica, replicação em máquinas de estados, etc.), usam um *serviço de eleição de líder* para prover os processos com a identidade de um líder, i.e., um processo correto no sistema. Tal serviço é tido como um oráculo distribuído que fornece a identidade de um processo considerado correto para todo nó que o invoca. Um oráculo da classe Ômega (Ω) pode fornecer informações equivocadas durante um período finito de tempo, mas a partir de um instante, todo processo correto recebe a identidade do mesmo processo correto. Diz-se então que todo processo no sistema confia naquele eleito como líder. A classe Ω reúne as condições minimais para se resolver o consenso [Chandra et al. 1996], um serviço fundamental que fatora a necessidade de acordo no sistema.

Este artigo propõe a implementação de um serviço de eleição de líder num sistema dinâmico assíncrono sujeito a falhas por parada (*crash*), em que os processos se comunicam através de uma memória constituída de registradores atômicos compartilhados.

O modelo de interação por memória compartilhada adequa-se bem a sistemas em que discos conectados em rede aceitam requisições de leitura e escrita diretamente de clientes. Esses discos constituem uma SAN (*storage area network*), que implementa uma abstração de memória compartilhada e tem se tornado cada vez mais frequente como mecanismo de tolerância a falhas. [Gafni and Lamport 2003] e [Guerraoui and Raynal 2007] apresentam algoritmos para tais sistemas baseados no detector de líder Ω. Adicionalmente, o desenvolvimento crescente de modernas arquiteturas multi-núcleo tem impulsionado estudos em sistemas de memória compartilhada assíncronos para os quais Ω tem sido aplicado como gerenciador de contenção, como em [Aguilera et al. 2003], [Fernández et al. 2010]. Apesar da importância de tais ambientes hoje, poucas são as propostas de modelos e protocolos para a implementação de Ω nessas circunstâncias.

Algumas abordagens para implementação de Ω no modelo de passagem de mensagens foram propostas. Inicialmente, elas consideravam um modelo com fortes requisitos de sincronia e confiabilidade dos canais [Chandra et al. 1996, Aguilera et al. 2001], mas, paulatinamente, foram sendo apresentadas soluções com requisitos cada vez mais fracos, considerando particularmente a existência de canais com requisitos temporais apenas para o líder [Aguilera et al. 2004, Malkhi et al. 2005, Hutle et al. 2009]. Entretanto, a totalidade de tais propostas são para o modelo clássico de sistemas distribuídos, cujo conjunto de processos é conhecido, além do número máximo f de falhas. Alguns poucos trabalhos existem para sistemas dinâmicos, cujo conjunto de processos é desconhecido e varia ao longo da execução; eles diferem no grau de conectividade e de conhecimento requerido do sistema [Jiménez et al. 2006, Fernández et al. 2006].

Para a comunicação por memória compartilhada, poucas são as propostas para Ω e a maioria delas considera sistemas estáticos [Guerraoui and Raynal 2006, Fernández et al. 2007, Fernández et al. 2010]. Apenas [Baldoni et al. 2009] incorpora alguns aspectos para tratar com o dinamismo do sistema, mas todas assumem canais com requisitos temporais. Uma estratégia, menos frequente, consiste em assumir um *padrão de mensagens* no sistema, ao invés de considerar limites para passos de processos ou transferência de mensagens. Protocolos baseados nesta abordagem não utilizam temporizadores e são por isso chamados *livres de tempo* (ou *time-free*). Nessa linha, alguns trabalhos surgiram para o modelo de passagem por mensagem [Mostefaoui et al. 2003, Arantes et al. 2013].

Até onde sabemos, até recentemente, não existia uma implementação de Ω para memória compartilhada assíncrona baseada em suposições livres de tempo. Em [Khouri and Greve 2014b], nós apresentamos uma primeira solução nesse sentido. Entretanto, o protocolo ali proposto não é eficiente, pois exige que todos os processos continuem a escrever na memória para sempre, mesmo após a eleição do líder. Os resultados ali apresentados servem como prova de conceito de que a eleição livre de tempo é possível.

Neste artigo, avançamos nessa compreensão do problema e apresentamos um protocolo Ω que, além de ser livre de tempo, é eficiente quanto às escritas. No caso, ele reune as condições necessárias para atingir escritas ótimas, que é quando, após a eleição,

somente o líder precisa continuar a escrever no seu registrador, enquanto os demais precisarão apenas ler. Isso ocorre sempre que os demais processos conseguem obter o último valor atualizado do registrador do líder. Ou seja, se todos os demais processos acessam a memória do líder somente depois que ele a atualizou. Com essa estratégia, tal como em [Aguilera et al. 2004, Malkhi et al. 2005, Hutle et al. 2009] que visavam optimalidade de sincronia, requerendo emissão de mensagens através de canais temporais apenas pelo líder, avançamos no estado da arte da implementação de Ω para memória compartilhada, requerendo que somente o líder continue a escrever na memória.

2. Modelo do Sistema

O sistema é composto por um conjunto infinito Π de processos que se comunicam através de uma memória compartilhada. Não fazemos suposições sobre a velocidade relativa dos processos, para dar passos ou acessar a memória, isto é, o sistema é assíncrono. Para facilitar a apresentação, assumimos a existência de um relógio global que não é acessível pelos processos, cuja faixa de ticks, $\mathcal{T}$, é o conjunto dos números naturais.

Consideramos o *modelo de chegada finita* [Aguilera 2004], isto é, o sistema tem um número infinito de processos que podem entrar e sair em instantes arbitrários, mas em cada execução r o número de participantes, n, é limitado, embora desconhecido. Os processos podem falhar parando arbitrariamente (*crashing*). Um processo *correto* comporta-se conforme a especificação, caso contrário, é *faltoso*. Cada processo p_i localiza-se num nó distinto e possui uma identidade distinta, i. O conjunto dessas identidades é totalmente ordenado. Denotamos um processo pelo seu nome: p_i, ou por sua identidade: i.

A memória compartilhada distribuída é uma abstração construída no topo de um sistema de passagem de mensagens que permite aos processos comunicarem-se invocando operações sobre objetos compartilhados. No nosso modelo, a memória consiste de arranjos de registradores atômicos multivalorados do tipo 1-*escritor-Múltiplos-leitores* ($1WMR$) que se comportam corretamente [Lamport 1986]. Um registrador é um tipo de objeto compartilhado que armazena um valor inteiro e provê duas operações básicas: $read$, que retorna o valor armazenado; e $write$, que escreve (armazena) um valor no registrador. Um registrador $R[i]$ pertence ao processo p_i, que é o seu único escritor, mas pode ser acessado por qualquer p_j para leitura.

Comportamento correto de um registrador significa que ele sempre pode executar uma leitura ou escrita e nunca corrompe o seu valor. Uma leitura sobre um *registrador atômico* que não é concorrente a uma escrita, obtém o último valor escrito no registrador. Quando uma leitura é concorrente a uma ou mais escritas sobre um registrador atômico pode retornar o valor antigo (escrito pela última operação de escrita não-concorrente à leitura), ou o valor sendo escrito por uma das escritas concorrentes [Lamport 1986].

Assumimos que a implementação da memória provê as operações $join()$ e $leave()$ que são invocadas pelos processos para entrar e sair, respectivamente, da computação. Um processo p_i que deseja entrar não têm conhecimento sobre o conjunto de participantes nem o estado dos objetos compartilhados. Assim, p_i invoca a função $join()$ que lhe provê uma identidade e o estado consistente do sistema; e informa aos demais nós sobre sua chegada. Dizemos que p_i está ativo a partir do retorno de $join()$ até que falhe ou deixe o sistema. Se um processo sai e volta para a computação, ele é considerado um novo processo e portanto terá uma nova identidade. Definimos assim o conjunto de processos ativos:

Definição 1 (ACTIVE) $ACTIVE(t) = \{p_a : p_a$ *está ativo no instante* $t\}$.

A despeito da entrada e saída de processos, a existência de um conjunto de processos estáveis num sistema dinâmico é necessária para que se possa realizar alguma computação útil. A noção de *estável* refere-se a um processo que entra no sistema e permanece ativo durante toda computação. O parâmetro α representa o limite inferior para a cardinalidade do conjunto de processos estáveis.

Definição 2 (STABLE) $STABLE = \{p_s : \exists t \in \mathcal{T}, \forall t' \geq t : p_s \in ATIVO(t')\}$

Para sistemas distribuídos clássicos de passagem de mensagem, o número de processos no sistema, n, e o limite superior para o número de falhas, f, desempenham papel fundamental no projeto de algoritmos, conquanto juntos, eles representam um limite inferior sobre o número de processos corretos para que o algoritmo convirja. Em um sistema dinâmico, onde nem n nem f são conhecidos, uma estratégia adotada é abstrair o par (n, f) através de um parâmetro α, que representa esse número mínimo de processos estáveis no sistema. É bem sabido que com passagem de mensagens, o limite para o número de processos faltosos é $n/2$, enquanto que com memória compartilhada, esse limite sobe para $n - 1$ [Lo and Hadzilacos 1994]. Considerando a possibilidade de $f \leq (n - 1)$ processos falharem, temos $\alpha \geq (n - (n - 1))$, então $\alpha \geq 1$. De modo semelhante, no modelo proposto, assumimos a existência de um parâmetro α, que define o número mínimo de processos $p_s \in STABLE$. Portanto, uma condição de progresso provida pelo modelo do sistema é dada pela Propriedade 1.

Propriedade 1 (Limite Inferior de $STABLE$**)** $|STABLE| \geq \alpha$.

2.1. Especificação de Ω

O uso de abstrações como detectores de falhas ou líderes possibilita a solução de certos problemas insolúveis em sistemas assíncronos sujeitos a falhas [Fischer et al. 1985]. Esses dispositivos funcionam como oráculos distribuídos que fornecem "dicas" sobre falhas de processos (ou líderes). Um serviço *detector de líder após um tempo* da classe Ω provê os processos com uma função *Leader()* que, quando invocada por um processo p_i, fornece a identidade de um processo p_j que o detector considera correto naquele instante. Após um tempo, essa identidade é única para todos os processos que invocam a função. Num contexto de sistema dinâmico, um detector Ω satisfaz à seguinte propriedade:

Liderança após um tempo (*eventual leadership*): existe um instante após o qual qualquer invocação de $Leader()$ por qualquer processo $p_i \in STABLE$ retorna a mesma identidade de processo $p_l \in STABLE$.

3. Eleição de Líder num Sistema Dinâmico de Memória Compartilhada

Para implementar Ω num sistema puramente assíncrono, é preciso enriquecer o sistema com suposições adicionais de sincronia que permitam a convergência. Em geral, essas suposições envolvem aspectos temporais, de modo que os protocolos utilizam temporizadores. Ao invés disso, introduzimos uma propriedade comportamental que reflete um padrão relativo aos acessos feitos à memória compartilhada.

3.1. Propriedade do Sistema

Considere um sistema em que cada processo ativo p_i escreve/atualiza periodicamente seu registrador $R[i]$ enquanto executa uma rodada de operações de leitura sobre os registradores $R[j]$ de p_j, a fim de obter os (novos) valores escritos. A rodada termina quando p_i tiver obtido um número "suficiente" de valores atualizados e, na sequência, p_i inicia uma nova rodada. Seja $U_i^{r_i}$ o conjunto dos primeiros α processos p_j cujos registradores foram encontrados atualizados por p_i na rodada r_i. A Propriedade 2 impõe a existência de um processo p_l que a partir de um instante t, está presente em $U_i^{r_i}, \forall r_i$, p_i é estável.

Propriedade 2 (Padrão de Acesso à Memória – PAM) *Existe um instante t e um processo $p_l \in STABLE$ tais que, $\forall p_i \in STABLE$, $\forall r_i$ iniciada após t : $p_l \in U_i^{r_i}$.*

Em última análise, a Propriedade 2 reflete alguma regularidade no comportamento do sistema, mesmo que ele se comporte de maneira assíncrona por um período arbitrário de tempo. É como se a partir de t, a comunicação entre cada processo p_i e p_l estivesse entre as (α) mais rápidas quando se considera a comunicação entre p_i e cada p_j.

3.2. Visão Geral do Algoritmo de Eleição de Líder

O Algoritmo é executado em paralelo por todos os processos ativos. Com o intuito de informar aos demais processos que está ativo, um processo p_x que se considera líder mantém-se incrementando um contador num registrador compartilhado $Alive[x]$. Por outro lado, periodicamente, cada p_i que considera p_x como líder, consulta o valor em $Alive[x]$ para saber se p_x progrediu. Se p_i não obtém um valor atualizado, antes de suspeitar de p_x, inicia uma rodada de leituras sobre cada $Alive[j]$ e espera encontrar pelo menos α registradores atualizados. Se $Alive[x]$ está entre os α primeiros encontrados, não é considerado suspeito, caso contrário, p_i suspeita de p_x e lhe aplica uma punição. Para esse fim, p_i mantém um arranjo de registradores – $Punishments$. Sempre que p_i suspeita de p_j, incrementa $Punishments[i][j]$. Como os processos compartilham a mesma visão de $Punishments$, p_i escolhe como líder o processo p_j para o qual $soma_i[j] = \sum_{\forall k} Punishments[k][j]$ é mínima. Se houver mais de uma soma com valor igual ao mínimo, o desempate será definido pela ordem lexicográfica, considerando as identidades dos processos. Dado o arranjo $soma_i$, a função $MinLex(soma_i)$ retorna x tal que:
$\forall x, y, (soma_i[x] = soma_i[y] \wedge x < y) \vee (soma_i[x] < soma_i[y])$.

No algoritmo são utilizados os seguintes registradores compartilhados:

- $Alive[i]$ – contador que indica o progresso de p_i; inicializado com 0.
- $Punishments[i][j]$ – contador de p_i para o número de punições aplicadas por p_i a p_j. No início, $Punishments[i][j] = 0, \forall i, j$. Quando um processo p_r entra depois, p_i inicializa $Punishments[i][r]$ com $Punishments[i][leader_i] + 1$.

Um processo p_i também utiliza as seguintes variáveis locais:

- $leader_i$ e ld_i – identidade do líder corrente em TASK 1 e TASK 2.
- $alive_i$ – controla o progresso de p_i cujo valor é escrito em $Alive[i]$. Inicializado com 0.
- $last\_leader_i$ e $last\_ld_i$ – identidade do líder da iteração anterior em TASK 1 e TASK 2.
- $last\_alive\_leader_i$ – último valor lido em $Alive[leader_i]$.
- $alive_i^j$ – variável auxiliar usada para leitura de $Alive[j]$.
- $last\_alive_i$ – vetor dinâmico que armazena o último valor lido em $Alive[1..]$. Cada entrada é inicializada com 0.

Algorithm 1 Leader_Election() - Part 1

```
    INITIALIZATION
 1: join();
 2: start TASK 0, TASK 1, TASK 2;
    TASK 0: INIT-REGS
    UPON REQUEST join() FROM p_j
 3: write(Punishments[i][j], Punishments[i][Leader()] + 1);
    TASK 1: ALIVENESS
 4: loop
 5:     leader_i ← Leader();
 6:     if (i = leader_i) then
 7:         alive_i ← alive_i + 1;
 8:         write (Alive[i], alive_i);
 9:         if (last_leader_i ≠ leader_i) then;
10:             last_leader_i ← leader_i;
11:     else
12:         read(Alive[leader_i], alive_leader_i);
13:         if (last_leader_i = leader_i) then
14:             if (last_alive_leader_i = alive_leader_i) then
15:                 alive_i ← alive_i + 1;
16:                 write (Alive[i], alive_i);
17:         else
18:             last_leader_i ← leader_i;
19:             last_alive_leader_i ← alive_leader_i;
```

- $soma_i$ – vetor dinâmico que armazena a soma dos contadores de punições de cada p_j.
- $updated_i$ – conjunto para controle dos processos que têm seu progresso verificado.
- pun_i – vetor dinâmico que armazena a quantidade de punições aplicadas por p_i a cada p_j. Cada entrada é inicializada com 0.

Notação: Dizemos que "p_i considera p_x como líder", se $leader_i = x$ (em TASK 1) ou $ld_i = x$ (em TASK 2). Dizemos que "p_x é demovido da posição de líder", se num instante t, $leader_i = x$ ($ld_i = x$) e na próxima atribuição de valor, $leader_i$ (ld_i) recebe $y \neq x$. Dizemos que "p_i pune p_j" quando p_i incrementa seu registrador $Punishments[i][j]$. "Atualizar um registrador", significa escrever no registrador seu valor corrente incrementado de 1. Um registrador $Alive[j]$ é considerado "atualizado" por p_i se ele obtém numa leitura um valor maior que o obtido na leitura imediatamente anterior.

3.3. Descrição do Algoritmo

Para um processo p_i entrar no sistema, invoca uma operação $join()$ que lhe provê: uma identidade; um estado consistente dos registradores; inicializa seus registradores compartilhados; e aloca espaço para seus arranjos dinâmicos locais. Os demais processos p_j, ao receberem uma requisição de $join()$, criam seus registradores $Punishments[j][i]$ devidamente inicializados e alocam espaço nos arranjos dinâmicos locais para acomodar p_i. Apenas após o retorno de $join()$, um processo p_i estará apto a participar da computação. Então p_i inicia três tarefas em paralelo.

Algorithm 2 Leader_Election() - Part 2

TASK 2: PUNISHMENT

20: **loop**
21: $ld_i \leftarrow Leader()$;
22: **if** $(ld_i \neq i)$ **then**
23: $updated_i \leftarrow \{i\}$;
24: **while** $((ld_i \notin updated_i) \wedge (|updated_i| < \alpha))$ **do**
25: **read** $(Alive[ld_i], alive\_ld_i)$;
26: **if** $(alive\_ld_i \neq last\_alive_i[ld_i])$ **then**
27: $last\_alive_i[ld_i] \leftarrow alive\_ld_i$;
28: $updated_i \leftarrow updated_i \cup ld_i$;
29: **else if** $(ld_i \notin updated_i)$ **then**
30: **for all** (j) **do**
31: **if** $(j \notin updated_i)$ **then**
32: **read** $(Alive[j], alive_i^j)$;
33: **if** $(alive_i^j \neq last\_alive_i[j])$ **then**
34: $last\_alive_i[j] \leftarrow alive_i^j$;
35: $updated_i \leftarrow updated_i \cup j$;
36: **if** $((|\ updated_i\ | < \alpha) \wedge (ld_i \notin updated_i))$ **then**
37: **if** $(leader_i \neq ld_i)$ **then** $ld_i \leftarrow leader_i$;
38: **if** $(ld_i \notin updated_i)$ **then**
39: **for** $(j \notin updated_i)$ **do**
40: $pun_i[j] \leftarrow pun_i[j] + 1$;
41: **write**$(Punishments[i][j], pun_i[j])$;

$Leader()$

42: **for all** (j) **do**
43: $soma_i[j] \leftarrow 0$;
44: **for all** (k) **do**
45: **read**$(Punishments[k][j], pun1_i)$;
46: $soma_i[j] \leftarrow soma_i[j] + pun1_i$;
47: $leader_i \leftarrow MinLex(soma_i[j])$;
48: **return** $leader_i$;

TASK 0: INIT-REGS – Ao perceber uma requisição de $join()$ de algum p_j, p_i inicializa o registrador $Punishments[i][j]$ com $Punishments[i][Leader()] + 1$ (linha 3).

TASK 1: ALIVENESS – Nesta tarefa, quando um processo p_i se considera líder, mantém-se atualizando $Alive[i]$, enquanto cada p_j verifica se $Alive[i]$ está atualizado. Consiste de um laço infinito no qual, inicialmente, cada processo p_i invoca a função $Leader()$ para definir $leader_i$ (linha 5). Para fazer certas verificações na iteração seguinte, p_i mantém a identidade do líder e de seu respectivo registrador $Alive$ em variáveis locais ($last\_leader_i$ – linhas 10 e 18) e ($last\_alive\_leader_i$ – linha 19). Enquanto p_i considera-se líder ($leader_i = i$), mantém-se atualizando $Alive[i]$ (linhas 6-8).

Um p_i que não se considera líder ($leader_i \neq i$),mantém-se verificando se $leader_i$ está vivo. Para isso, p_i lê $Alive[leader_i]$ (linha 12) e verifica se o líder corrente é o mesmo da iteração anterior (linha 13). Se isso ocorre e $Alive[leader_i]$ não foi atualizado (linha

14), p_i atualiza seu $Alive[i]$ (linhas 15-16). Senão, p_i apenas guarda a identidade do líder e o valor de seu registrador (linhas 18 e 19). Isso significa que é possível que apenas o líder eleito permaneça escrevendo em $Alive$ pra sempre.

TASK 2: PUNISHMENT – Aqui também são utilizadas variáveis locais, $last\_alive_i[j]$, para guardar o valor lido em $Alive[j]$ e possibilitar a verificação do progresso de p_j (linhas 27, 34). O objetivo da tarefa é punir processos suspeitos, mas isso só ocorre se $leader_i$ é um dos suspeitos. Então, no início do laço, p_i define um líder, ld_i, invocando $Leader()$ (linha 21). Nenhum processo suspeita de si mesmo. Então, enquanto p_i se considera líder, apenas invoca $Leader()$ (linhas 21-22). Já um processo $p_i \neq ld_i$, inicializa o conjunto $updated_i$ com sua própria identidade (linha 23) e entra num laço **while** (linha 24) onde permanece até que uma das condições seja satisfeita – *Condição 1*: $ld_i \in updated_i$, o que significa que $Alive[ld_i]$ foi atualizado; ou *Condição 2*: $\mid updated_i \mid \geq \alpha$, que implica que pelo menos α processos p_j atualizaram $Alive[j]$. Como o algoritmo foi desenhado para sistemas assíncronos estendidos com as Propriedades 1 e 2, essas condições garantem que, ao cabo de um tempo, algum processo p_l sempre estará em $updated_i, \forall i$.

No **while**, p_i lê $Alive[ld_i]$ (linha 25). Se obtém um valor atualizado, guarda-o em $last\_alive_i[ld_i]$ e inclui ld_i em $updated_i$ (linhas 26-28). Dessa forma, a *Condição 1* é imediatamente satisfeita. Então, p_i sai do laço, avalia o predicado da linha 38: *falso* e não pune qualquer p_j. Se, no entanto, p_i não obtém um valor atualizado, entra num laço **for** (linhas 30-35) onde lê cada $Alive[j]$ e, caso obtenha um valor atualizado, inclui j em $updated_i$. Após o **for**, se a *Condição 1* é satisfeita, p_i sai do **while** e vai para a próxima iteração do **loop**. Se a *Condição 2* é satisfeita e $ld_i \notin updated_i$, p_i pune cada $p_j \notin updated_i$ e vai para a próxima iteração do **loop** (linhas 38-41). Para evitar que um processo permaneça bloqueado no **while**, após cada iteração do **for**, p_i verifica se o líder que ele obteve na última invocação a $Leader()$ em TASK 1 – $leader_i$, é o mesmo sendo considerado em TASK 1 – ld_i. Se não for, ele altera ld_i para receber $leader_i$ (linhas 36-**??**). Quando a Propriedade 2 se verifica, algum p_l eleito como líder por todo p_i não será mais punido, portanto nenhuma escrita mais será efetuada sobre $Punishments$.

A função Leader() – Retorna uma identidade de processo que, do ponto de vista de p_i, é correto e considerado líder por ele. Para essa escolha, a soma das punições atribuídas a cada p_j é acumulada em $soma_i[j]$ (linhas 42-46). É escolhido como líder o processo com menor soma, utilizando a identidade como desempate (linhas 47-48).

4. Corretude do Algoritmo

Esta seção apresenta um resumo da prova de que os processos estáveis elegem um único processo estável – o líder, ao cabo de um tempo. Uma versão completa da prova pode ser obtida em [Khouri and Greve 2014a], que devido à restrições de espaço não pode ser aqui detalhada.

Lema 1 *Existe um instante t e um processo $p_l \in STABLE$ tais que após t, p_l não será jamais punido por qualquer processo p_i.*

Prova. As punições ocorrem na TASK 2 (linhas 38-41). Se $p_i \in STABLE$, de acordo com a Propriedade 2, existem um processo p_l e um instante t', tais que em toda rodada de leituras sobre os registradores $Alive$, iniciada por p_i após t', resultará na inclusão de p_l em $updated_i$ (linhas 25-28 ou 30-35), antes que p_i saia do **while** da linha 24. Logo, a partir de um instante $t > t'$, p_l não será punido por qualquer p_i. $_{Lema1}\square$

Lema 2 *Nenhum processo p_i fica permanentemente bloqueado executando o Algoritmo Leader_Election().*

Prova. O único trecho do algoritmo em que um processo pode ficar bloqueado é o laço **while** da TASK 2 (linhas 24-**??**), o qual só é executado por processos que não se consideram líderes (linha 22). Para mostrar que um processo não fica bloqueado no laço, devemos mostrar que pelo menos uma das condições é satisfeita: *Condição 1*: $ld_i \in updated_i$; ou *Condição 2*: $\mid updated_i \mid \geq \alpha$ (linha 24). Considere os seguintes casos possíveis em que um processo estável p_i que considera p_x como líder executa o **while** da TASK 2.

Caso 1: p_x é estável e considera-se líder. Logo, atualiza $Alive[x]$ (linha 8). Ao cabo de um tempo, p_i obtém $Alive[x]$ atualizado (linha 25 ou 32) e a *Condição 1* é satisfeita.

Caso 2: p_x é faltoso e falha em um instante t quando pelo menos α processos estáveis p_j o consideram como líder na TASK 1 (linha 5). Então, uma vez que nenhum p_j obtém $Alive[x]$ atualizado, p_j atualiza $Alive[j]$ (linhas 12-16). Na TASK 2, p_i inclui cada p_j em $updated_i$ (linhas 30-35) e após um tempo, a *Condição 2* é satisfeita.

Caso 3: p_x é faltoso e falha no instante t, quando menos de α processos estáveis p_j o consideram como líder na TASK 1 (linha 5). Como p_i não inclui p_x em $updated_i$ (linhas 28 ou 35), a *Condição 1* não é satisfeita. Então, devemos mostrar que p_i obtém α registradores $Alive$ atualizados. Uma vez que $\mid STABLE \mid \geq \alpha$ (Propriedade 1), se menos de α processos estáveis p_j consideram p_x como líder, sabemos que existe algum processo p_k tal que $leader_k = y, y \neq x$. Considere o seguinte cenário, sem perda de generalidade. (I) No instante t, $\exists p_j : leader_j = x$ e $\exists p_k : leader_k = y$. (II) No intervalo de tempo $I = [t', t''], t' < t < t$“, as únicas operações de escrita sobre $Punishments$ em andamento, são operações sobre $Punishments[-][x]$ e $Punishments[-][y]$. (III) Existem operações de leitura sobre esses mesmos registradores executadas por p_j e p_k concorrentes às escritas (linhas 42-46). Daí temos que, (A) após um tempo, se nenhuma nova escrita é requisitada sobre esses registradores, todo p_h que invoca $Leader()$ na TASK 1 (linha 5), retorna z. Se z é correto, atualiza $Alive[z]$ (A1); Senão, todo p_j correto atualiza $Alive[j]$ (A2) (linhas 12-16). Em TASK 2 p_i obtém essas atualizações e a *Condição 1* ou *Condição 2* é satisfeita (De (A1) ou (A2)). Por outro lado, (B) para que novas punições ocorram, também é preciso que pelo menos α processos p_h suspeitem de p_z e atualizem $Alive[h]$ na TASK 1 (linhas 12-16). Assim, p_i obtém α $Alive[h]$ atualizados, inclui cada p_h em $updated_i$, e a *Condição 2* é satisfeita.

Caso 4: p_x é estável, mas não se considera líder. Então, p_x não atualiza $Alive[x]$ na linha 8 mas pode ser que o faça na linha 16, se suspeita de seu líder. Se isso ocorre, a *Condição 1* é satisfeita. Caso contrário, se pelo menos α p_j consideram p_x como líder, tal como nos *Casos 2* e *3*, a *Condição 2* é satisfeita. Senão, temos os vários desdobramentos do *Caso 3* e, ao cabo de um tempo, uma das duas condições é satisfeita. $_{Lema2}\square$

Lema 3 *Existe um instante t e um processo $p_l \in STABLE$ tais que $\forall t' \geq t$, $\forall i$, $leader_i = ld_i = p_l$.*

Prova. Em um instante t, p_i define um processo p_x como líder, se ao invocar $Leader()$, obtém um estado de $Punishments$ tal que $MinLex(\sum_{\forall k} Punishments[k][j]) = x$.

Pelo Lema 1, algum processo p_l (que satisfaz a Propriedade 2) não é mais punido a partir de um instante t. Isto é, seus contadores de punições permanecem inalterados, enquanto que os demais podem crescer. Assim, após um tempo, p_l terá o menor nível de punições e será eleito líder. Como não é mais punido, também não é demovido. $_{Lema3}\square$

Lema 4 *Se um processo p_x é considerado líder por algum processo p_i e falha no instante t, então, existe um instante $t' > t$, em que p_x é demovido da posição de líder, e a partir de um instante $t'' \geq t'$, p_x não será considerado líder por qualquer processo.*

Prova. Temos a considerar os seguintes casos.

Caso 1: No instante t, pelo menos α p_j consideram p_x como líder na TASK 1, quando p_x falha. A prova segue dos argumentos de prova do *Caso 2* – Lema 2.

Caso 2: Menos de α p_j consideram p_x como líder na TASK 1 (linha 5), quando p_x falha, no instante t. A prova segue da prova do *Caso 3* – Lema 2.

Teorema 1 *Existe um instante após o qual algum $p_l \in STABLE$ será eleito permanentemente líder.*

Prova. Segue direto dos Lemas 3, 1, 2 e 4. $_{Teorema1}\square$

5. Complexidade

O algoritmo usa n registradores $Alive$ e n^2 registradores $Punishments$, isto é, $O(n^2)$ registradores $1WMR$, onde n é o número de processos que entra no sistema. Uma vez que o número de rodadas não é limitado, o tamanho dos registradores $Alive$ é ilimitado também. No entanto, os registradores $Punishments$ são limitados, já que a partir do instante em que PAM é satisfeita, nenhum processo mais é punido.

Com relação à complexidade de passo, temos que, inicialmente, um processo p_i invoca a função $Leader()$ para definir um líder. Nesta função, nenhuma escrita é executada, e são realizadas $[n(r)]^2$ operações de leitura sobre $Punishments$, onde $n(r)$ é o número de processos que já entraram até a rodada r. Em TASK 0, p_i executa 1 escrita cada vez que um novo processo p_r entra no sistema. Nenhuma leitura é feita nesta tarefa. Em TASK 1, p_i executa n^2 operações de leitura (em $Leader()$) e então: (1) se $leader_i = i$, p_i executa 1 escrita em $Alive[i]$; (2) se $leader_i = j \neq i$, p_i executa 1 leitura em $Alive[j]$ e pode ter que executar 1 escrita ou não. Isto é, p_i executa, no máximo, 1 escrita.

Em TASK 2, p_i executa n^2 operações de leitura (em $Leader()$), e então: (1) se $leader_i = i$, p_i não executa qualquer acesso à memória compartilhada; (2) se $leader_i = j$, p_i executa (a) 1 leitura em $Alive[j]$ e nenhuma escrita, caso obtenha $Alive[j]$ atualizado nesta leitura; ou (b) pelo menos $n(r)$ leituras sobre $Alive[-]$, se obtém $Alive[j]$ atualizado em alguma dessas leituras, e nenhuma escrita; ou (c) no pior caso, p_i suspeita de p_j e executa, pelo menos $n(r)$ leituras sobre $Alive[-]$ e tantas punições quantos forem os registradores encontrados desatualizados, isto é, até $n(r) - \alpha$ escritas em $Punishments$.

Portanto, a partir do instante em que a propriedade 2 é satisfeita, e considerando que a chegada de processos cessou, em uma rodada, um processo p_i executa, no pior caso, $O(n^2)$ leituras e 1 escrita. Focando num sistema onde leituras são locais [Baldoni et al. 2009] (ou mesmo mais baratas que escritas), o custo real do algoritmo deve-se às operações de escrita. Por esse motivo, a abordagem utilizada no projeto do algoritmo foi a redução das escritas. Considerando execuções *bem-comportadas*, nas quais o registrador do líder é sempre encontrado atualizado, apenas o líder eleito permanece escrevendo para sempre na memória compartilhada. O mesmo ocorre no caso especial em que $\alpha = 1$. Nessas situações, o comportamento do algoritmo é ótimo, já que em qualquer algoritmo de Ω para o modelo proposto, pelo menos um processo, o líder eleito, p_l, precisa informar seu progresso pra sempre [Fernández et al. 2010].

6. Trabalhos Relacionados

A Tabela 1 resume as principais abordagens para eleição eventual de líder e implementação de Ω em sistemas assíncronos propostas na literatura. Podemos distinguir duas grandes categorias: (i) os que estendem o sistema assíncrono com requisitos temporais (a grande maioria) e são considerados (*timer-based*) e (ii) os que não usam o tempo, mas padrão de troca de mensagens ou de acesso à memória (*message-pattern* ou *time-free*). Como a Tabela 1 mostra, a grande maioria considera o modelo de passagem de mensagens e poucos são para o de memória compartilhada.

Abordagem	Comunicação	Π	Conectividade	Propriedade do Modelo
[Chandra et al. 1996]	pass. de mensagens	conhecido	grafo completo	sistema parcialmente síncrono
$\exists p$ *eventually accessible* [Aguilera et al. 2001]	pass. de mensagens	conhecido	grafo completo	$\exists p$: todos os canais bidirecionais de/para p são *eventually timely*
$\exists p$ que é $\diamondsuit f$-*source* [Aguilera et al. 2004]	pass. de mensagens	conhecido	grafo completo	$\exists p$: f canais de saída de p são *eventually timely*
$\exists p$ que é $\diamondsuit f$-*accessible* [Malkhi et al. 2005]	pass. de mensagens	conhecido	grafo completo	$\exists p, Q$: *round-trip* de p para $\forall q \in Q$, é *eventually timely*
$\exists p$ que é *eventually moving* f-*source* [Hutle et al. 2009]	pass. de mensagens	conhecido	grafo completo	$\exists p, Q$: canal de saída de p para $\forall q \in Q$, é *eventually timely*
[Jiménez et al. 2006]	pass. de mensagens	desconhecido dinâmico	nós corretos conectados	$\exists p$ correto: $\forall q$ correto, pode ser atingido de p por canal *eventually timely*
[Fernández et al. 2006]	pass. de mensagens	desconhecido dinâmico	nós corretos conectados	$\exists p$ correto e $>(n$-$f)$-$\alpha+1$ processos corretos alcançáveis de p por canais *eventually timely*
[Guerraoui and Raynal 2006]	memória compartilhada	conhecido	-	sistema síncrono
temporizadores "bem comportados" [Fernández et al. 2010]	memória compartilhada	conhecido	-	$\exists p$: cujos acessos à memória compartilhada são *eventually timely*
mecanismo *query-response* (*time-free*) [Mostefaoui et al. 2003]	pass. de mensagens	conhecido	grafo completo	$\exists t, p, Q$: a partir de t, $q \in Q$ recebe uma resposta de p à suas *queries* entre as $(n-f)$ primeiras.
mecanismo *query-response* (*time-free*) [Arantes et al. 2013]	pass. de mensagens	desconhecido dinâmico	grafo completo	$\exists t, p, Q$: a partir de t, $q \in Q$ recebe uma resposta de p à suas *queries* entre as α primeiras.
padrão de acesso à memória [Khouri and Greve 2014b]	memória compartilhada	desconhecido dinâmico	-	$\exists t, p_l$: a partir de t, $Alive[l]$ está entre os α primeiros registradores atualizados

Tabela 1. Abordagens para Implementação de Ω

6.1. Soluções Clássicas para Modelo de Sistemas Estáticos

A busca por requisitos mínimos relativos ao grau de sincronia e confiabilidade dos canais e seu grau de conectividade foram fruto de estudo de diversos trabalhos para o modelo de passagem de mensagem. As 7 entradas iniciais da Tabela 1 têm tal abordagem. Os primeiros estudos de detectores de líder consideravam uma rede de comunicação completamente conectada com todos os canais confiáveis e síncronos a partir de um determinado instante [Chandra et al. 1996]. Já [Aguilera et al. 2001] considera que existe pelo menos um processo correto p e um instante t, tais que a partir de t, todos os canais bidirecionais entre p e todos os outros processos são síncronos (*eventually timely*). [Aguilera et al. 2004] mostram que Ω pode ser implementado em sistemas com falhas por parada, cujos canais sejam *fair-lossy* (permitem perdas equitáveis), desde que pelo menos um processo correto p possua f canais de saída que sejam *eventually timely* (p é dito ser um $\diamondsuit f\text{-}source$).

Malkhi et al. (2005) usam uma estratégia diferente: a noção de *eventually* f-*acessibility*. Supõe-se um instante t, um processo p e um conjunto Q (variável com o

tempo) de f processos, tais que, $\forall t' \geq t$, p troca mensagens com cada q dentro do limite de tempo δ (relativo ao *round-trip* do canal), isto é, p é $\diamondsuit f$*-accessible*. não é fixo, isto é, pode variar com o tempo. Uma suposição ainda mais fraca é a proposta em [Hutle et al. 2009]: existe um processo correto p, a saber, um *eventually moving f-source*, tal que, após um tempo, cada mensagem que ele envia é recebida em tempo (*timely*) por um conjunto Q de f processos que pode ser diferente em instantes distintos.

Para memória compartilhada, há poucas propostas para Ω e a maioria delas considera sistemas estáticos. [Guerraoui and Raynal 2006] implementam um protocolo com registradores regulares mas diferente do nosso trabalho, consideram um sistema estático, síncrono após um tempo. [Fernández et al. 2007], [Fernández et al. 2010], consideram um sistema assíncrono com registradores atômicos compartilhados; mas, ao contrário da nossa proposta, supõem um ambiente estático, além de canais síncronos.

6.2. Soluções Recentes para Modelo de Sistemas Dinâmicos

Em sistemas dinâmicos, há uma imprevisibilidade sobre o conjunto de participantes da computação, bem como sobre os canais de comunicação. [Jiménez et al. 2006] propõem um protocolo para ambiente dinâmico com troca de mensagens, supondo canais *eventually timely*. Já [Fernández et al. 2006] apresentam dois protocolos. No primeiro eles consideram que os processos não conhecem o número de participantes n nem o limite para o número de falhas f. Como na nossa abordagem, cada processo só conhece sua própria identidade e um limite inferior α sobre o número de processos corretos. Diferentemente da nossa proposta, entretanto, os processos se comunicam por troca de mensagens e o algoritmo, baseado em *timeouts*, exige a existência de alguns canais *eventually timely*.

Em um sistema de memória compartilhada cujos participantes são desconhecidos, um processo que chega, naturalmente, precisa inteirar-se do estado do sistema antes de participar efetivamente. [Baldoni et al. 2009] propõem uma abstração de registrador regular compartilhado no topo de um sistema dinâmico de passagem de mensagens. Quando um processo p_i deseja entrar, invoca uma operação $join()$, que lhe permite obter um estado consistente do registrador. De modo semelhante, o nosso protocolo admite o uso de uma operação $join()$ que se encarrega de criar registradores devidamente inicializados.

Abordagens Livres de Tempos Baseadas em Padrão de Mensagens. [Mostefaoui et al. 2003, Mostefaoui et al. 2006] introduziram uma abordagem baseada num mecanismo *query-response* que assume que a resposta de algum p_i às últimas requisições de p_j está sempre entre as primeiras $n - f$ respostas recebidas por cada p_j. A intuição por trás da suposição é a de que, mesmo que o sistema não exiba propriedades síncronas, ele pode apresentar alguma regularidade de comportamento que se traduza numa *sincronia lógica* capaz de contornar a impossibilidade de conversão do algoritmo.

A partir de uma suposição semelhante, [Arantes et al. 2013] propõem um modelo para implementação de Ω num sistema dinâmico de passagem de mensagens que considera a mobilidade dos nós. Supondo um padrão de mensagens adaptado a sistemas dinâmicos, segundo o qual os processos envolvidos pertencem a um conjunto de nós *estáveis*, eles utilizam o limite inferior α sobre o número de processos estáveis para controlar o número de respostas consideradas entre as primeiras. Essa constante α abstrai todos os pares (n, f) tradicionalmente consideradas, tal que $n - f \geq \alpha$. Dessa forma, um protocolo baseado em α, funciona para qualquer par (n, f).

Mais recentemente, [Khouri and Greve 2014b] apresentaram um protocolo para implementar Ω num modelo de memória compartilhada que, semelhantemente ao proposto aqui, tem por base um padrão de acesso à memória compartilhada. Até onde sabemos, não existe outra implementação de Ω para memória compartilhada baseada em suposições livre de tempo. Nesse sentido, os dois trabalhos são pioneiros.

Entretanto, [Khouri and Greve 2014b] apresentam um protocolo pouco eficiente, cujos processos (todos) continuam a escrever nos registradores para sempre, mesmo após a eleição do líder. O protocolo aqui apresentado avança na obtenção de uma solução ótima para as escritas, reunindo condições para que somente o líder, após um tempo, continue a escrever no seu registrador. Com tal estratégia, semelhantemente às soluções [Aguilera et al. 2004, Malkhi et al. 2005, Hutle et al. 2009] para passagem de mensagens, que visavam optimalidade na quantidade de canais síncronos e de processos emissores de mensagens, avançamos no estado da arte da implementação de Ω para memória compartilhada. Em ambos os casos, o objetivo de fazer com que somente o líder atue (emita mensagens ou escreva no registrador, após um tempo) é atingido.

7. Considerações Finais

Este artigo avança na compreensão do problema de eleição de líder em memória compartilhada, considerando um sistema dinâmico sem requisitos temporais. Em resumo, as suas principais contribuições são: (i) A definição de um modelo de sistema dinâmico sujeito a uma propriedade livre de tempo baseada num padrão de acesso à memória, que permite a implementação de um serviço de líder numa memória compartilhada assíncrona, onde o conjunto de participantes é desconhecido; (ii) Um protocolo que implementa um serviço de líder da classe Ω adequado ao modelo proposto e que visa optimalidade na escrita dos registradores. O sistema dinâmico considera a existência de um conjunto de processos estáveis (entram no sistema e permanecem ativos sem falhar ou sair), cuja cardinalidade mínima é α. O algoritmo proposto tem resiliência ótima, já que mantém a corretude se ao menos um único processo estável permanece no sistema ($\alpha \geq 1$). Como trabalho futuro pretende-se avaliar a possibilidade de uso de registradores regulares ao invés de atômicos, já que os primeiros são mais fracos.

Referências

Aguilera, M., Delporte-Gallet, C., Fauconnier, H., and Toueg, S. (2004). Communication-efficient leader election and consensus with limited link synchrony. In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, pages 328–337, New York, NY, USA. ACM.

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

Aguilera, M. K., Delporte-Gallet, C., Fauconnier, H., and Toueg, S. (2001). Stable leader election. In Welch, J., editor, *Distributed Computing*, volume 2180 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin Heidelberg.

Aguilera, M. K., Englert, B., and Gafni, E. (2003). On using network attached disks as shared memory. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, PODC '03, pages 315–324, New York, NY, USA. ACM.

Arantes, L., Greve, F., Sens, P., and Simon, V. (2013). Eventual leader election in envolving mobile networks. In *Proceedings of the 17th International Conference on Principles of Distributed Systems*, OPODIS '13, Berlin, Heidelberg. Springer-Verlag.

Baldoni, R., Bonomi, S., Kermarrec, A.-M., and Raynal, M. (2009). Implementing a register in a dynamic distributed system. In *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, pages 639–647.

Chandra, T. D., Hadzilacos, V., and Toueg, S. (1996). The weakest failure detector for solving consensus. *Journal of the ACM*, 43(4):685–722.

Fernández, A., Jiménez, E., and Raynal, M. (2006). Eventual leader election with weak assumptions on initial knowledge, communication reliability, and synchrony. In *In DSN*, pages 166–175.

Fernández, A., Jiménez, E., and Raynal, M. (2007). Electing an eventual leader in an asynchronous shared memory system. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 399–408.

Fernández, A., Jiménez, E., Raynal, M., and Trédan, G. (2010). A timing assumption and two t-resilient protocols for implementing an eventual leader service inÂ asynchronous shared memory systems. *Algorithmica*, 56(4):550–576.

Fischer, M. J., Lynch, N. A., and Paterson, M. D. (1985). Impossibility of distributed consensus with one faulty process. *Journal of ACM*, 32(2):374–382.

Gafni, E. and Lamport, L. (2003). Disk paxos. *Distrib. Comput.*, 16(1):1–20.

Guerraoui, R. and Raynal, M. (2006). A leader election protocol for eventually synchronous shared memory systems. In *Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006. The Fourth IEEE Workshop on*, pages 6 pp.–.

Guerraoui, R. and Raynal, M. (2007). The alpha of indulgent consensus. *Comput. J.*, 50(1):53–67.

Hutle, M., Malkhi, D., Schmid, U., and Zhou, L. (2009). Chasing the weakest system model for implementing ω and consensus. *IEEE Transactions on Dependable and Secure Computing*, 6(4):269–281.

Jiménez, E., Arévalo, S., and Fernández, A. (2006). Implementing unreliable failure detectors with unknown membership. *Information Processing Letters*, 100(2):60 – 63.

Khouri, C. and Greve, F. (2014a). Eleicão assíncrona de líder em memória compartilhada dinâmica. In *Relatório Técnico*, number 12, Salvador, BA. DCC/UFBA.

Khouri, C. and Greve, F. (2014b). Servico de líder em sistemas dinâmicos com memória compartilhada. XIV WTF-SBRC, Porto Alegre, RS, Brasil. SBC.

Lamport, L. (1986). On interprocess communication. *Distributed Computing*, 1(2):77–101.

Lo, W.-K. and Hadzilacos, V. (1994). Using failure detectors to solve consensus in asynchronous shared-memory systems (extended abstract). In *Proceedings of the 8th International Workshop on Distributed Algorithms*, WDAG '94, pages 280–295, London, UK. Springer-Verlag.

Malkhi, D., Oprea, F., and Zhou, L. (2005). ω meets paxos: Leader election and stability without eventual timely links. In *Proceedings of the 19th International Conference on Distributed Computing*, DISC'05, pages 199–213, Berlin, Heidelberg. Springer-Verlag.

Mostefaoui, A., Mourgaya, E., and Raynal, M. (2003). M.: Asynchronous implementation of failure detectors. In *In: Proc. International IEEE Conference on Dependable Systems and Networks (DSN'03*, pages 351–360.

Mostefaoui, A., Mourgaya, E., Raynal, M., and Travers, C. (2006). A time-free assumption to implement eventual leadership. *Parallel Processing Letters*, 16(02):189–207.

Flexible Content Placement in Cache Networks using Reinforced Counters

Guilherme Domingues[1],
Edmundo de Souza e Silva[1], **Rosa M. M. Leao**[1], **Daniel S. Menasche**[1]

[1]Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, RJ – Brazil

{guilhdom, edmundo, rosam, sadoc}@land.ufrj.br

Abstract. *In this paper we study the problem of content placement in a cache network. We consider a network where routing of requests is based on random walks. Content placement is done using a novel mechanism referred to as "reinforced counters". To each content we associate a counter, which is incremented every time the content is requested, and which is decremented at a fixed rate. We model and analyze this mechanism, tuning its parameters so as to achieve desired performance goals for a single cache or for a cache network. We also show that the optimal static content placement, without reinforced counters, is NP hard under different design goals.*

1. Introduction

In today's Internet the demand for multimedia files and the sizes of these files are steadily increasing. The popularity of Youtube, Dropbox and Netflix, to name a few, motivate researchers to seek for novel cost-effective content dissemination solutions.

Caching is one of the most classical solutions to increase the load supported by computer systems. In essence, caching consists of transparently storing data in a way that future requests can be served faster. Caching in the realm of standalone computer architectures received significant attention from the research community since the early sixties, and web-caching has also been studied for at least one decade. The objects of study of this work, in contrast, are cache networks, which were proposed and started to receive focus much more recently [Zhang and Carofiglio 2012, Jacobson et al. 2009].

Cache networks comprise two main features, caching and routing, both performed by the same core component: a cache router. In a cache network, content traverses the network from sources to destinations, but can in turn be stored in caches strategically placed on top of the routers. The *data plane* is responsible for transmitting contents whereas the *control plane* is responsible for the routing of requests. The system considered in this paper is illustrated in Figure 1. A request for file F is routed using random walks through the caches. Every time the request hits a cache, the cache selects uniformly at random one of the links that are incident to the cache and uses it to forward the request. Once the request reaches a cache where the content is stored, the content is transferred to the requester through the data plane. We assume that all contents are stored in at least one cache.

In this paper we study the problem of content placement in cache networks. We pose the following questions: (1) How to store and evict contents from a cache in a flexible and scalable manner? (2) How to efficiently and distributedly place content in a cache network?

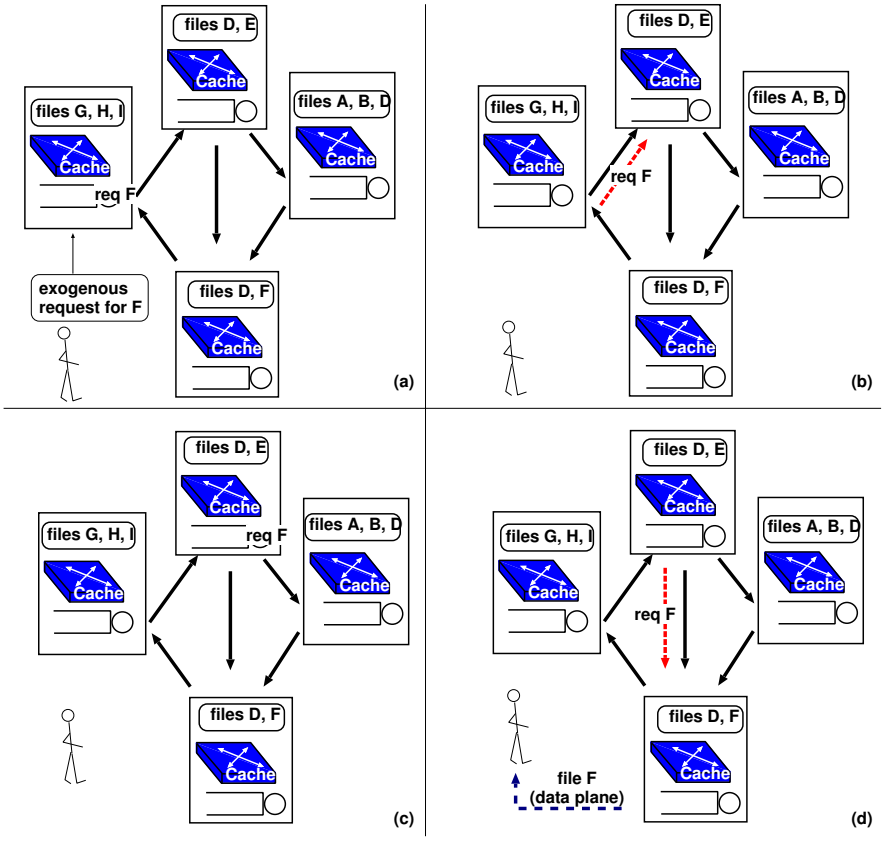


Figura 1. Illustration of how system works: (a) **a request is placed in cache 1;** (b) **the request is routed to cache 2;** (c) **the request is processed in cache 2;** (d) **the request is routed to cache 4. The file is transferred through the data plane.**

Content placement is done using a novel mechanism referred to as "reinforced counters" [Domingues et al. 2013, Domingues et al. 2014]. In [Domingues et al. 2013, Domingues et al. 2014] we have introduced RCs as an analytically-tractable solution for content placement. In this paper, we present a detailed analysis of the solution, and extend with features such as hysteresis. We then show how to use RCs to control different metrics that affect overall system performance. To each content we associate a counter, which is incremented every time the content is requested, and which is decremented at a fixed rate. We model and analyze this mechanism, tuning its parameters so as to achieve desired performance goals for both a single cache and for a cache network.

Each cache in a cache network has a given service capacity. The service capacity of a cache is related to the time it takes to 1) find content in the cache, 2) return that content to the user, through the data plane, in case of a cache hit and 3) route the request to another cache, through the control plane, in case of a cache miss. Note that in the two latter cases, the service capacity of a cache network accounts for network delays due to transmission and queueing. A cache is stable if, for a given workload, the queue of pending requests does not grow unboundedly with respect to time. A cache network is stable if, for a given workload, all its caches are stable. We show that the optimal static content placement, without reinforced counters, is NP hard under different design goals accounting for stability and content availability.

To summarize, in (partially) answering the questions above we make the following contributions

Analytical model of RC: we introduce reinforced counters as a way to flexibly store and evict contents from a cache, showing that they are amenable to analytical study and optimal tuning;

Optimal content placement: we propose a new formulation of the optimal content placement in cache networks accounting for stability. Then, we show that static content placement, without reinforced counters, is NP hard, which motivates the use of reinforced counters or variants in a network setting.

The rest of this paper is organized as follows. Section §2 studies the single cache police based on the reinforced counters. In §3 we address the problem of multiple caches and we show that under different setting the problem at hand is NP hard. In §4 we present related work, and §5 concludes.

2. Single Cache: Reinforced Counters and Flexible Content Placement

In this section, we study placement and eviction policies for a single cache under the assumption that the dynamics of each content is decoupled from the others. Decoupled content dynamics yields tractable analysis and can be used to approximate the performance of systems with fixed capacity. They are also of interest in the context of DNS caches and the novel Amazon ElastiCache system (`http://aws.amazon.com/elasticache/`).

In the policies to be introduced in this section, the expected number of items in the system can be controlled. Let π_{up} be the probability that each content is stored in the cache. Given a collection of N contents the expected number of stored contents is $N\pi_{up}$, which can be controlled or bounded according to users needs. In what follows, we consider the problem of controlling π_{up} using *reinforced counters* (Section 2.1). In Section 2.2 we illustrate how the mechanism works through some simple numerical examples. Then, in Section 2.3 we extend the reinforced counters to allow for hysteresis, showing the benefits of hysteresis for higher predictability and reduced chances of content removal before full download.

2.1. Reinforced counter with a single threshold

To each content we associate a reinforced counter [Domingues et al. 2013, Domingues et al. 2014], which is increased by one every time the content is requested, and is decreased by one as a timer ticks. Let K be the reinforced counter eviction threshold. When the counter is incremented from K to $K+1$, the content is stored. Whenever the counter is decremented from $K+1$ to K the content is evicted. The timer ticks every $1/\mu$ seconds. Henceforth, we assume that the time between ticks is exponentially distributed. This mechanism is similar to TTL caches, employed by DNS and web-caching systems [Berger et al. 2014, Fofack et al. 2012], noting that content is inserted in the cache only after the threshold K is surpassed. The advantages of using RCs over existing mechanisms will be shown in this section.

We assume that the behavior of each of the contents is decoupled. This assumption yields an analytically-tractable model and is of interest in a number of settings, including a cloud where resources are virtually unlimited.

In the cloud, we may assume that storage is infinite. Users might incur costs which are proportional to the space used, but the storage space itself is unlimited. Under

the mean field approximation [Fofack et al. 2012], it has been shown that decoupling the dynamics of multiple contents can lead to reasonable approximations to the content hit probabilities. Instead of considering that the cache has a fixed capacity, it is assumed that there are constraints on the expected number of items in the cache.

Note that under the mean field approximation the cache capacity design problem can be compared to the problem of calculating the capacity of communication lines in a telephone network. When capacity is finite, one can take advantage of statistical multiplexing either by evicting items from the cache when (a) an overflow occurs or (b) when the item expire. This is particularly relevant in the context of time-to-live caches (such as those used by the DNS system) where entries need to be renewed from time to time to avoid staleness.

Our goal in the remainder of this section is to show the advantages of having a content placement mechanism with two associated knobs, μ and K, allowing for the user to fine tune both the fraction of time in which the content is in the cache (steady state metric) while at the same time controlling the mean time between content insertions (or, equivalently, controlling the rate at which content is evicted or brought back into cache) so as to avoid that content is replaced too fast and content starvation (that is, content is never included into cache or removed from it during a finite but large time interval). The timer tick rate μ and the reinforced counter threshold K can be tuned so as to adjust the long term fraction of time in which the content is stored in the cache and to guarantee that the mean time between content eviction and content reinsertion into the cache is bounded.

In what follows, we assume that requests for a given content arrive according to a Poisson process with rate λ. λ is also referred to as the content popularity. (Recall that the behavior of each content is decoupled for others.) Figure 2 is useful to illustrate the different intervals of the cache content replacement and the notation we use. The blue (red) intervals in the figure indicate that the content is stored (or not) in cache. If content is not in cache it is brought into cache when a new request for it arrives and the reinforced counter is at the threshold K. On the other hand, if the value of the reinforced counter is at $K + 1$ and the counter ticks, the content is removed from cache. Note that we adopt the same assumption as in [Fofack et al. 2012]: the insertion and eviction of a content is not influenced by other contents in the same cache. As mentioned above, fixed storage in cache is modeled by considering the expected number of contents into the cache.

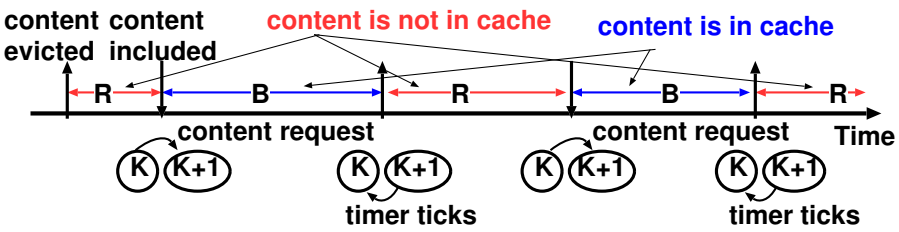


Figura 2. Request counter and notation.

Let π_{up} be the fraction of time in which the content is in the cache. Each cache content alternates from periods of inclusion and exclusion from cache (see Figure 2), and

we have (renewal theory),

$$\pi_{up} = \frac{E[B]}{E[R] + E[B]} \tag{1}$$

where $E[R]$ is the mean time that a content takes to return to the cache once it is evicted and $E[B]$ is the mean time that the content remains in the cache after insertion. It should be clear that the reinforced counter mechanism can be modeled as an M/M/1 queueing system, with state equal to the value of the reinforced counter. Then,

$$\pi_{up} = \sum_{i=K+1}^{\infty} (1-\rho)\rho^i = \rho^{K+1} \tag{2}$$

where $\rho = \lambda/\mu$.

Let $\gamma(K, \mu)$ be the rate at which content enters the cache. Due to flow balance, in steady state, $\gamma(K, \mu)$ equals the rate at which content leaves the cache,

$$\gamma(K, \mu) = \mu\rho^{K+1}(1-\rho) = \lambda\rho^K(1-\rho) = \frac{1}{E[B] + E[R]} \tag{3}$$

where the last equality can be easily inferred from Figure 2 (renewal arguments).

$E[B]$ can be calculated from first passage time arguments, that is the time it takes from the system to return to state K (eviction) once it is brought into system (state $(K+1)$). From the M/M/1 model, it can also be calculated by the busy period of an M/G/1 queue, which equals

$$E[B] = 1/(\mu - \lambda). \tag{4}$$

Then, from (4) and (1)

$$\pi_{up} = \frac{1/(\mu-\lambda)}{1/(\mu-\lambda) + E[R]}. \tag{5}$$

Once the content is evicted the mean time for it to return to the cache is given by

$$E[R] = (1 - \pi_{up})/(\pi_{up}(\mu - \lambda)) \tag{6}$$

Given a fixed π_{up}, it is possible to write ρ, μ, $E[R]$ and γ as a function of K,

$$\begin{aligned}
\rho &= \pi_{up}^{1/(K+1)} && (7)\\
\mu &= \lambda\pi_{up}^{-1/(K+1)} && (8)\\
E[R] &= (1-\pi_{up})/(\pi_{up}(\lambda(\pi_{up})^{-1/(K+1)} - \lambda)) && (9)\\
\gamma &= \lambda\pi_{up}((\pi_{up})^{-1/(K+1)} - 1) && (10)
\end{aligned}$$

Note that there is a tradeoff in the choice of K, as increasing the value of K reduces the rate at which content is inserted into the cache (see equation (3)), which in turn reduces the steady state costs to download the content from external sources. The larger the value of K, the smaller the steady state rate at which the content enters and leaves the cache. But increasing the value of K, also increases the mean time for the content to be reinserted into the cache once the content is evicted. The larger the value

of K, the longer the requesters for a given content will have to wait in order to be able to download the content from the cache after it is evicted (see equation (9)).

Let K_{max} be the maximum value allowed for K. Motivated by the tradeoff above, given a fixed value of π_{up} we consider the following optimization problem,

$$\begin{aligned} \min_{K} \quad & \psi(K) = \alpha\gamma + \beta E[R] && (11)\\ \text{such that} \quad & && (12)\\ K \quad & \leq \quad K_{max} && (13)\\ \pi_{up} \quad & = \quad \rho^{K+1} && (14)\\ \rho \quad & = \quad \lambda/(\lambda\pi_{up}^{-1/(K+1)}) && (15) \end{aligned}$$

α and β are used to control the relevance of long term and short term dynamics. The long term dynamics reflect the behavior of the system after a long period of time, during which the rate at which content enters the cache is given by $\mu\rho^{K+1}(1-\rho)$. The short term dynamics reflect the behavior of the system during a shorter period of time, during which one wants to guarantee that the mean time it takes for the content to return to the cache is not too large. We should keep in mind that we choose π_{up} to satisfy cache capacity limitations and system performance.

Substituting (7)-(10) into (11), the objective function is given by,

$$\psi(K) = \alpha\Big[\lambda\pi_{up}((\pi_{up})^{-1/(K+1)} - 1)\Big] + \beta(1-\pi_{up})\frac{1}{\pi_{up}\lambda}\Big[\frac{1}{(\pi_{up})^{-1/(K+1)} - 1}\Big] \quad (16)$$

The value of $E[R]$ must be bounded so as to avoid starvation, as formalized in the propositions below.

Proposition 2.1. *If $\beta = 0$, the optimal strategy consists of setting $K = K_{max}$. If $K_{max} = \infty$, it will take infinite time for the content to be reinserted in the cache once it is evicted for the first time.*

Proof: The objective function is given by

$$\psi(K) = \alpha\lambda\pi_{up}((\pi_{up})^{-1/(K+1)} - 1) \quad (17)$$

The derivative of the expression above with respect to K is

$$\frac{d\psi(K)}{dK} = \alpha\frac{\lambda\log(\pi_{up})\pi_{up}}{(K+1)^2(\pi_{up})^{1/(K+1)}} \quad (18)$$

which is readily verified to be always negative. Therefore, the minimum is reached when $K = K_{max}$. When $K = \infty$ it follows from (8) that μ tends to 0. The mean time for reinsertion of the content in the cache is given by (9) which grows unboundedly as μ tends to 0. □

Proposition 2.2. *If $\beta > 0$ the optimization problem* (11)-(15) *admits a unique minimum $K^\star$.*

The proof is omitted for conciseness.

2.2. Illustrative Example

Next, we consider an illustrative example to show the tradeoff in the choice of K. Let $\pi_{up} = 0.9$ and $\alpha = \beta = 1$. Figure 3 shows how cost first decreases and then increases, as K increases. The optimal is reached for $K = 10$. At that point, we have $E[R] = 0.31$ and $\gamma = 0.32$.

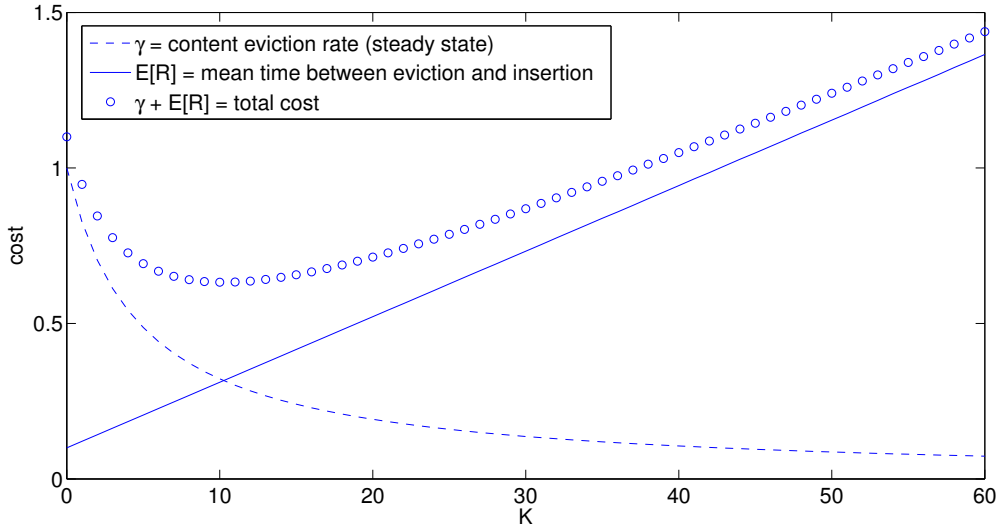


Figura 3. $\pi_{up} = 0.9$, $\alpha = \beta = 1$

Remark 1: Note that in problem (11)-(15) we do not set a hard constraint on $E[R]$. Alternatively, we could add such a constraint, $E[R] \leq R^{\star}$. The solution of the modified problem is either the solution of the problem without the constraint (in case the constraint is inactive) or the value of K which yields $E[R] = R^{\star}$ (in case the constraint is active).

Remark 2: If follows from Markov inequality that the solution of the problem (11)-(15) naturally yields a bound on the probability $P(R > r)$, i.e., $P(R > r) < E[R]/r$. In the numerical example above, when K is optimally set we have that the probability that the the content is not reinserted into cache after 3.1 units of time following an eviction is $P(R > 0.31 \times 10) < 0.1$. In the numerical results we present latter $P(R > r)$ is calculated exactly from the model.

Remark 3: We assume that K can take real values. If K is not an integer, we can always randomize between the two closest integers when deciding whether to store or not the content.

2.3. Reinforced counter with hysteresis

In this section we generalize the reinforced counter to allow for a third control knob K_h as follows. The counter is incremented at each arrival request for a content and is decremented at rate $1/\mu$. In addition, the content is included into cache when the value of the reinforced counter is incremented to $K + 1$, as in previous section. However, content is not removed from cache when the counter value is decremented from $K + 1$ to K. Instead, content remains in cache until the counter reaches the (new) threshold K_h. Figure

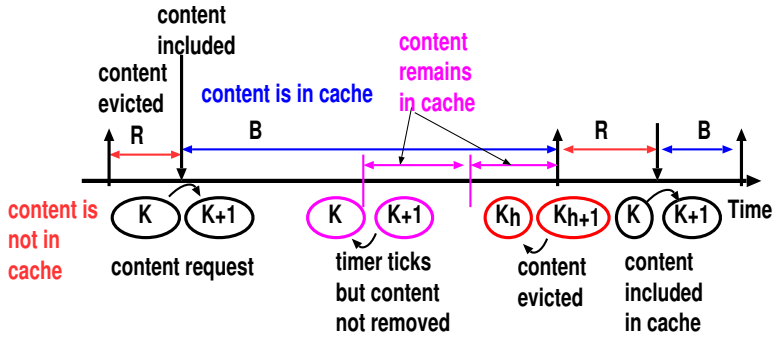


Figura 4. Reinforced counter with hysteresis.

4 illustrates the behavior of the new reinforced counter. From the figure, we observe that the purple intervals correspond to the values of the reinforced counter between K and K_h and the content is in cache. The bottom of Figure 4 shows a trajectory of the reinforced counter.

At first glance, it seems that only the intervals where the content is in cache are affected by this new mechanism (the blue intervals) but not the intervals where content is absent (the red intervals). However, this is not true and both intervals are affected. In what follows, we show how to calculate the expected values of $E[B]$ and $E[R]$ and how the measures of interest are affected by this new mechanism. We also show the advantages of the reinforced counter with hysteresis. For that, we refer to Figure 5 that shows the Markov chain for the hysteresis counter.

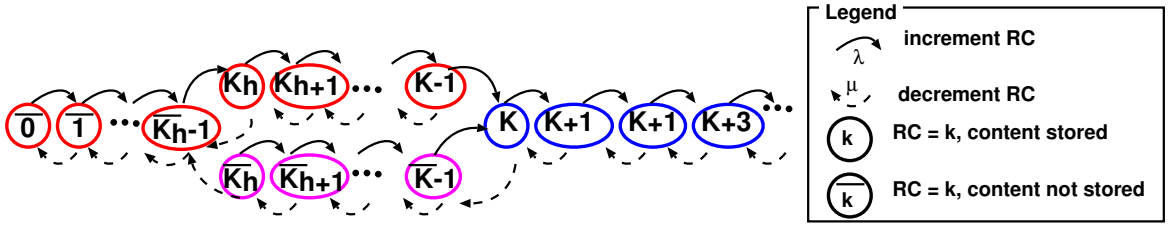


Figura 5. Reinforced counter with hysteresis: state transition diagram.

Let $\nu(i+1) = E[B]$ and $\xi(i+1) = E[R]$ both when $K - K_h = i$. Note that $\nu(1)$ and $\xi(1)$ are the values of $E[B]$ and $E[R]$ for a cache with no hysteresis. $\nu(1)$ is obtained from equations (1), (2) and (9). $\xi(1) = \left(\sum_{j=0}^{K} 1/\rho^j\right) / (\lambda/(\lambda+\mu))$.

Proposition 2.3. *$\nu(i+1)$ and $\xi(i+1)$ can be obtained by the following recursions:*

$$\nu(i) = \tfrac{1}{\mu} + \nu(i-1) + \rho\nu(1) \qquad i \geq 2 \tag{19}$$

$$\xi(i) = \tfrac{1}{\lambda} + \xi(i-1) + \tfrac{1}{\rho}\xi(1) \qquad 2 \leq i \leq K - K_h + 1 \tag{20}$$

Proof: The proof follows from renewal arguments, and is omitted due to space limitations. □

It is important to note that explicit expressions for $\nu(i)$ and $\xi(i)$ can be obtained as a function of λ, μ and K but details are omitted since the recursion above suffices to explain our comments.

Suppose λ and π_{up} are given and we obtain K and μ, for instance from the optimization problem in the previous section. We allow K_h to vary from $K_h = K$ (that is reinforced counter without hysteresis) to $K_h = 0$.

Proposition 2.4. *As K_h decreases, the rate $\gamma(K, \mu)$ at which content enters the cache decreases.*

Proof: From Proposition 2.3, it is not difficult to see that both $E[B]$ and $E[R]$ increase with K_h. Since $\gamma(K, \mu) = 1/(E[B] + E[R])$ (equation (3)) the result follows. □

Proposition 2.4 shows that, from an initial value of π_{up}, if we fix the parameters λ and K, the rate at which content is replaced (both included and removed from cache) decreases by using the hysteresis mechanism, which is good to lower costs as explained in the previous section. However, π_{up} also varies. As a consequence of Proposition 2.3 we can show that π_{up} is reduced. This is not obvious since $E[B]$ increases. But, by adjusting the knob μ, π_{up} can be maintained constant while γ is reduced when the hysteresis schema is used. The proof of this last result is omitted but it follows from Proposition 2.3. The numerical results presented in the Section 2.4 corroborate the claim.

There are additional advantages of the hysteresis mechanism. First note that, in the previous sections, we assumed that file download times are negligible. However, when a user requests for a content it is important that the whole file remains stored at the cache not only until this user finishes downloading but also while other users are downloading the same content from that cache. Hysteresis is helpful to prevent the file from being removed before its download is concluded by all requesters. In Section 2.4 we show that hysteresis increases the probability that a content remains in cache for at least some time t after it is cached. Note that this is an additional (transient) performance measure and it differs from the γ metric (steady state rate). As our numerical results show, by adjusting K_h we can improve both steady state and transient metrics.

Our numerical results also show that hysteresis reduces the coefficient of variation of the time that content resides in the cache (B). This is important for cache capacity planning with multiple contents, since more predictable systems are usually easier to design and control.

2.4. Numerical Results

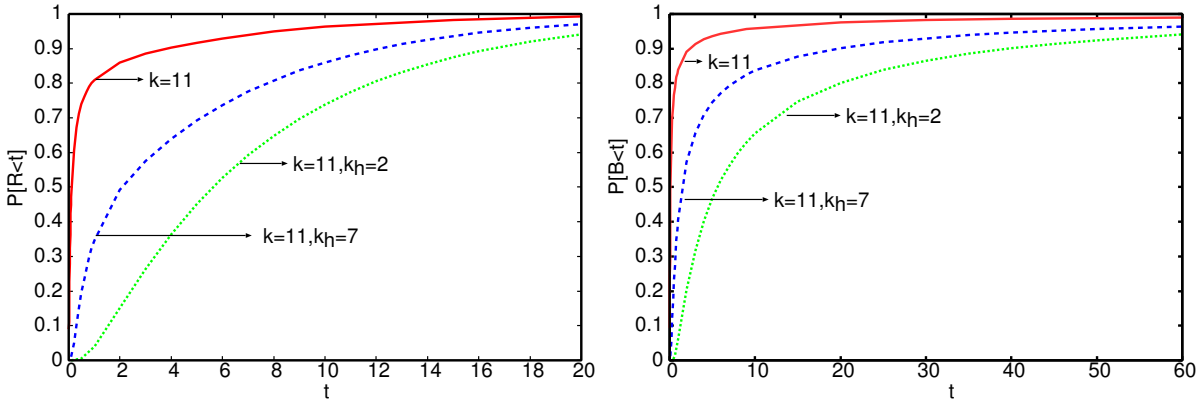
In this section we show some numerical results obtained for the models: the reinforced counter with a single threshold (K) and the reinforced counter with two thresholds (K,K_h). Three performance measures were used to analyze the models: the rate at which content enters the cache (γ), the cumulative distribution of the time the content takes to return to the cache (R) and the cumulative distribution of the time the content remains in the cache after insertion (B).

Three scenarios were evaluated: (a) the reinforced counter has a single threshold $K = 11$, (b) the reinforced counter has two thresholds $K = 11$ and $K_h = 7$, and (c) the reinforced counter has two thresholds $K = 11$ and $K_h = 2$. For each scenario, we consider the same value of π_{up}, λ, and K.

The value of γ for scenario (a) is 0.24, for (b) is 0.07 and for (c) is 0.04. We note that the rate at which content enters or leaves the cache decreases as the value of K_h decreases. This is one of the advantages of introducing a third control knob K_h.

Figures 6(a) and 6(b) show the cumulative distribution of R and B. We note that the reinforced counter with hysteresis allow to control the probability distribution of R and B. In Figure 6(a), consider for example $t = 4$. If we set $K_h = 2$, we have $P[R < 4] = 0.35$ and if $K_h = 7$, then $P[R < 4] = 0.65$. As the value of K_h increases, the probability of R be less than a certain value of t increases. This behavior can also be observed for the distribution of B. On the other hand, if we consider the model with a single threshold we can not control the distribution of R and B. In Figure 6(a), $P[R < 4] = 0.9$.

Another advantage of the reinforced counter with hysteresis, is that the coefficient of variation of R and B, decreases with the value of K_h, which means that the dispersion of the distribution of R and B also decreases. The values obtained for the coefficient of variation of B for each scenario are: (a) 1.6, (b) 1.3 and (c) 1.1.



(a) Cumulative Distribution of R

(b) Cumulative Distribution of B

Figura 6. Cumulative Distribution of the time the content takes to return/remains in the cache.

3. Multiple Caches

In the previous section we argued that we can decouple the cache storage dynamics of each content and study each content in isolation. Roughly, decoupling is a consequence of statistical multiplexing, for content placement policies that limit the amount of time a single content is cached, independently of the other requested contents. This is the case of the *reinforced counter* policies we study. We showed the advantages of these policies and the flexibility they bring to the design of caches.

In this section we address the problem of multiple caches in a network. The main objectives are: (a) to formulate the problem of content placement at a cache network; (b) to show that if we do not use a decoupling policy such as those we study in this paper, the optimal content placement problem is NP hard. This last result emphasizes the importance of employing a cache placement mechanism like the reinforced counters.

3.1. Model and Problem Formulation

Let $\mathcal{F}$ and $\mathcal{C}$ be the set of files and caches in the system. There are $F = |\mathcal{F}|$ files and $C = |\mathcal{C}|$ caches in the system. File f has size t_f, $1 \leq f \leq F$ and cache i has (storage) capacity s_i, $1 \leq i \leq C$. Users issue exogenous requests for files. At each cache i, $1 \leq i \leq C$, exogenous requests for file j arrive at rate λ_{ij}. We assume that cache i has service capacity η_i requests/s. Once a request arrives at a cache,

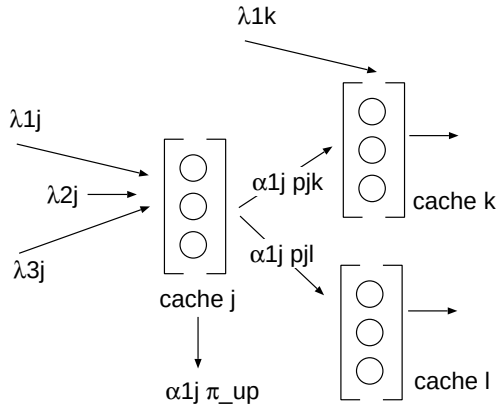


Figura 7. Illustration of cache network

1. **in case of a cache hit**, the content is immediately transferred back to the requester through the data plane. Our model is easily adapted to account for the delay in searching for a file in the cache, but for the sake of presentation conciseness we assume zero searching cost in this paper;
2. **in case of a cache miss**, the request is added to a pool of requests to be serviced. The request is processed and transferred to one of the outgoing links, selected uniformly at random.

Let A_{ij} be a variable that indicates if file j is available at cache i. From the results of previous section, $A_{ij} = 1$ with probability π_{up} for content j at cache i. Therefore, we can compute $P[A_{ij} = 1]$ using the derived results. In addition, we may also study cases in which content is statically placed into caches, i.e., the content in the caches is not replaced. This corresponds, for instance, to a system where the replacements of files from the caches occur at a much coarser granularity than the requests. In this case, $A_{ij} = 1$ if file j is at cache i and 0 otherwise.

Let M_{hi} be the adjacency matrix between caches, i.e., $M_{hi} = 1$ if there is a direct path from cache h to cache i and 0 otherwise.

Let d_h be the outgoing degree of node h. Let p_{hi} be the probability that a request from cache h is routed to cache i. In this paper, we consider random walk routing ([Domingues et al. 2013, Domingues et al. 2014]), i.e., $p_{hi} = 1/|d_h|$ if $M_{hi} > 0$ and 0 otherwise.

Figure 7 illustrates the problem in the case where we have a cache network comprised of 3 nodes, j, k and l, and 3 contents, indexed by 1, 2 and 3. In the Figure, the exogenous arrival rates $\lambda_j^{(1)}, \lambda_j^{(2)}, \ldots$ of contents 1, 2 and 3 to cache j are made explicit. Let $\pi_{up}^{(c)}(i)$ the designed parameter π_{up} in previous sections, but for cache i and content (c). Let $\alpha_j^{(c)}$ represents the total input rate of requests to content (c) arriving at cache j (exogenous arrivals to cache j plus those requests coming from other caches). A fraction $\alpha_j^{(c)}\pi_{up}^{(c)}(j)$ is immediately served, and fractions $\alpha_j^{(c)}(1-\pi_{up}^{(c)}(j))p_{jk}$ and $\alpha_j^{(c)}(1-\pi_{up}^{(c)}(j))p_{jl}$ are transferred to caches k and l, respectively.

In the remainder of this paper, we will assume that requests for files that are cached at i, $1 \leq i \leq C$, are immediately processed at i, and we will be concerned with transfers

that occur due to misses.

$$\alpha_j^{(c)} = \lambda_j^{(c)} + \sum_{k=1,k\neq j}^{C} \alpha_k^{(c)}(1 - \pi_{up}^{(c)}(k))p_{kj} \tag{21}$$

We can compute the mean response time from the expected total number of requests in the system. Using Little's law, and recalling that the processing rate is η_i we have that the average number of requests at cache i is $E[N_i] = \alpha_i/\eta_i$. Then, the expected total number of requests in the system is $E[N] = \sum_{i=1}^{C} E[N_i]$ and the mean response time, $E[T] = E[N]/\Lambda$ where Λ is the sum of the exogenous arrival rates to all caches.

Definition 3.1: The cache network is stable if and only if $\eta_i > \alpha_i$ for all i, $1 \leq i \leq C$. $\diamond$

Until this point, we focused on the use of reinforced counters for content placement. In the remainder of the paper, we show that without reinforced counters a natural statement of optimal content placement yields an NP hard problem.

Definition 3.2: The optimal content placement problem consists of finding the mapping $A : \mathcal{F} \to \mathcal{C}$ such that

1. the cache network is stable
2. $\sum_{f=1}^{F} 1_{A_{if}=1} t_f \leq s_i$, for $i = 1, \ldots F$
3. the arrival rates of requests that require processing at the control plane, $\sum_{i=1}^{C} \alpha_i$, is minimized

$\diamond$

3.2. Static Content Placement in Cache Networks is NP Hard

We show that the problem of static optimal content placement in cache networks is NP hard. This result is interesting because it shows the importance of using a cache replacement policy like those we study in this paper. Recall that, by using the reinforcement policy, we can optimize the counter parameters to satisfy a given probability of finding a content in cache for each content in isolation. These probabilities are in turn obtained to satisfy a given cache capacity constraint, from statistical multiplexing arguments. (the problem is identical to sizing a telephone network and one can use the results of that area.)

First, we consider the case where different files have different sizes. In this case, it is possible to show that the problem is NP hard even if we need to make placement decisions at a single cache. To this aim, we can consider a mapping from KNAPSACK (the proof is omitted due to space constraints). Then, we consider the simpler case wherein all files have the same size. In this case, if we need to make placement decisions at a single cache, the problem can be solved using a greedy strategy. Nevertheless, we show that the problem is still NP hard if we need to make placement decisions in at least two caches.

Theorem 3.1. *The optimal content placement problem is NP hard.*

Proof: We refer to FEASIBLECACHE as the problem of deciding if a given cache network admits a feasible content placement. Note that the problem of finding the optimal placement, i.e. the PLACEMENTPROBLEM, must be harder than FEASIBLECACHE, since solving the optimization problem yields a solution to FEASIBLECACHE. We proceed with a Turing reduction from the PARTITION problem to the FEASIBLECACHE problem. The complete proof is omitted due to space limitations. $\square$

4. Related Work

The literature on caching [Che et al. 2009] and content placement [Presti et al. 2005], and its relations to networking [Neves et al. 2014], database systems and operating systems [Nelson et al. 1988] is vast. Nevertheless, the study of cache networks, which encompass networks where routing and caching decisions are taken together at devices which work as routers and caches is scarce [Rosensweig et al. 2010]. In this work, we present a systematic analysis of cache networks using reinforced counters and indicating their applicability both in the single cache as well as multiple cache settings.

Cache networks play a key role in content centric networks (CCNs) [Jacobson et al. 2009, Koponen et al. 2007]. CCNs are emerging as one of the potential architectures for the future Internet. In CCNs, content rather than hosts is addressed. The *glue* that binds CCNs are the content chunks rather than the IP packets [Jacobson et al. 2009].

Using caches to reduce publishing costs can be helpful for publishers that cannot afford staying online all the time, or need to limit the bandwidth consumed to replicate content. With caches, content can *persist* in the network even in the absence of a publisher [Koponen et al. 2007]. In case of intermittent publishers, content availability can only be guaranteed in case replicas of the content are placed in some of the caches.

Approximate and exact analysis of single caches and cache networks has been studied for decades [Berger et al. 2014, Bianchi et al. 2013, Fofack et al. 2012]. Approximate analysis is usually carried out using mean field approaches [Bianchi et al. 2013] or other asymptotic techniques [Jelenković 1999], whereas exact analysis is performed accounting for Markov chain properties [Gast and Houdt 2015] or assuming uncoupled content dynamics [Fofack et al. 2012]. In this paper, we consider reinforced counters as the placement algorithms, which are at the same time amenable to analytical study and of practical interest in the context of DNS systems or the novel Amazon ElastiCache [Amazon 2014, Berger et al. 2014]. Cache policies are traditionally devised to avoid staleness of content and/or improve system efficiency. The analysis presented in this paper shows that reinforced counters can be used to target these two goals.

5. Conclusion and Future Work

In this paper we have studied the problem of scalable and efficient content placement in cache networks. We analyzed an opportunistic caching policy using reinforced counters [Domingues et al. 2013] as an efficient mechanism for content placement and proposed an extension to the basic scheme (reinforced counters with hysteresis). Using reinforced counters, we indicated how to tune simple knobs in order to reach the optimal placement. We then showed some of the properties of the mechanism and the optimal content placement problem. Finally, using the queueing-network model we propose for a cache network, we showed how to calculate the overall expected delay for a request to obtain a content. In addition, we showed that obtaining the optimal solution without reinforced counters is an NP hard problem.

This work is a first step towards the characterization of efficient, scalable and tractable content placement in cache networks. Future work consists of studying how to distributedly tune the caches, allowing the presence of custodians that store permanent copies of the files. We also aim at simulating the proposed solutions under real topologies [Jannibelli 2014].

Referências

Amazon (2014). Amazon elasticache. `aws.amazon.com/elasticache/`.

Berger, D. S., Gland, P., Singla, S., and Ciucu, F. (2014). Exact analysis of ttl cache networks: The case of caching policies driven by stopping times. *arXiv preprint arXiv:1402.5987*.

Bianchi, G., Melazzi, N. B., Caponi, A., and Detti, A. (2013). A general, tractable and accurate model for a cascade of caches. *arXiv preprint arXiv:1309.0718*.

Che, H., Tung, Y., and Wang, Z. (2009). Hierarchical web caching systems: Modeling, design and experimental results. In *JSAC*.

Domingues, G., de Souza e Silva, E., Leão, R. M. M., and Menasché, D. S. (2013). Enabling information centric networks through opportunistic search, routing and caching. In *SBRC*, pages 135–148.

Domingues, G., de Souza e Silva, E., Leão, R. M. M., and Menasché, D. S. (2014). A name resolution assisted icn design, supported by opportunistic search, routing and caching policies. In *Workshop em Desempenho de Sistemas Computacionais e de Comunicação (Wperformance)*.

Fofack, N. C., Nain, P., Neglia, G., and Towsley, D. (2012). Analysis of ttl-based cache networks. In *VALUETOOLS*, pages 1–10. IEEE.

Gast, N. and Houdt, B. V. (2015). Transient and steady-state regime of a family of list-based cache replacement algorithms. In *SIGMETRICS*.

Jacobson, V., Smetters, D. K., Briggs, N. H., Thornton, J. D., Plass, M. F., and Braynard, R. L. (2009). Networking named content. In *CONEXT*.

Jannibelli, D. V. (2014). Simulador para um modelo de busca oportunística de conteúdos em redes orientadas a informação. Master's thesis, COPPE UFRJ, Rio de Janeiro.

Jelenković, P. R. (1999). Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities. *Annals of Applied Probability*, pages 430–464.

Koponen, T., Chawla, M., Chun, B., Ermolinskiy, A., Kim, K., Shenker, S., and Stoica, I. (2007). A data-oriented (and beyond) network architecture. In *SIGCOMM*.

Nelson, M. N., Welch, B. B., and Ousterhout, J. K. (1988). Caching in the sprite network file system. *ACM Transactions on Computer Systems (TOCS)*, 6(1):134–154.

Neves, T., Ochi, L. S., and Albuquerque, C. (2014). A new hybrid heuristic for replica placement and request distribution in content distribution networks. *Optimization Letters*, pages 1–16.

Presti, F. L., Bartolini, N., and Petrioli, C. (2005). Redirection in content delivery networks. In *ICC*.

Rosensweig, E. J., Kurose, J., and Towsley, D. (2010). Approximate models for general cache networks. In *INFOCOM*.

Zhang, L. and Carofiglio, G. (2012). 1st workshop on emerging design choices in name-oriented networking. In *INFOCOM*.

Gerência Distribuída de Controle de Uso para PaaS Utilizando Gabarito de Regras e Credenciais de Autorização

Maicon Stihler, Altair O. Santin, Arlindo L. Marcon Jr.

Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba – PR – Brasil

Abstract. *The popularization of PaaS environments presents challenges to traditional authorization models for controlling resource usage. This article describes a hybrid authorization model suitable for PaaS environments, using rule templates, authorization credentials and local derivation of individual policies. In addition, we present the usage control mechanisms that eliminate the main disadvantages of policy models based on provisioning and outsourcing. The work also shows the technical details of the implementation of a prototype with its performance evaluation results.*

Resumo. *A popularização de ambientes de PaaS apresenta desafios para o controle de uso dos recursos e modelos de autorização tradicionais. Este artigo descreve um modelo de autorização híbrido apropriado para ambientes PaaS, que usa gabaritos de regras, credenciais de autorização e derivação local de políticas individuais. Além disso, são apresentados os mecanismos de controle de uso que eliminam as principais desvantagens dos modelos de políticas baseados em provisioning e outsourcing. O trabalho mostra também os detalhes técnicos da implementação de um protótipo juntamente com resultados da avaliação de desempenho do mesmo.*

1. Introdução

O termo Computação em Nuvem [Hayes 2008] se refere à utilização de diversas tecnologias para criar um ambiente dinâmico, onde organizações podem implementar seus próprios serviços usando infraestrutura de TI sob demanda. Isto é, as organizações podem criar serviços que se adaptam dinamicamente às suas necessidades, através do emprego de uma infraestrutura elástica (redimensionável). Tais recursos podem ser utilizados sem a necessidade de instalação e configuração de equipamentos e softwares especializados, sendo acessíveis através de qualquer computador conectado à Internet.

Para que isso seja possível, a computação em nuvem está organizada em três modelos de serviço bem definidos, de acordo com o nível de abstração da funcionalidade oferecida [Mell 2009]. São eles: Infraestrutura como serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). O modelo de IaaS oferece recursos computacionais básicos (e.g. armazenamento, processamento e redes) como serviço, permitindo que as organizações instalem e executem softwares à sua escolha sobre uma infraestrutura virtual. Os ambientes de PaaS, oferecem serviços intermediários (e.g. segurança, gerenciamento de usuários) que podem ser utilizados para o desenvolvimento e execução de aplicações, ocultando a infraestrutura subjacente. No modelo de SaaS, os usuários

utilizam aplicações prontas disponibilizadas pelo provedor. Esses modelos são conceitualmente independentes entre si, sendo possível, mas não mandatório, utilizar os serviços de um modelo (e.g. IaaS) para criar um ambiente de maior abstração (e.g. PaaS ou SaaS).

De acordo com [Takabi et al. 2010], por atender um público heterogêneo, com serviços diversos e com requerimentos diferentes, a computação em nuvem exige políticas de controle de acesso que sejam de granularidade fina. Os mecanismos de controle de acesso devem ser capazes de capturar a dinamicidade da nuvem, utilizando informações contextuais, de atributos e credenciais. Adicionalmente, é desejável que a nuvem ofereça controles que permitam capturar aspectos relevantes de nível de serviço (SLA), enquanto oferecem meios de gerenciamento amigáveis e eficientes.

Para atender à essa demanda por dinamismo, uma das tendências tem sido a utilização do modelo de controle de uso, UCON$_{ABC}$ [Park and Sandhu 2004], por sua capacidade de refletir mudanças (e.g. em atributos de usuários, objetos, ambiente) em tempo de execução. Pesquisas anteriores para aplicação de UCON$_{ABC}$ em nuvens computacionais ([Marcon Jr. et al. 2014, Lazouski et al. 2012, Danwei Chen 2009]), têm utilizado abordagens centralizadoras, isto é, baseadas no modelo de *outsourcing* e, por isso, herdam suas principais desvantagens: os custos de comunicação de rede (*overhead*), ponto único de falha no monitor de referência e baixa escalabilidade. Em [Marcon Jr. et al. 2014] é proposta a utilização de *espaços de tuplas* para minimizar esses problemas, porém, tal abordagem não é baseada em padrões abertos e requer a implementação de mecanismos complexos, como demonstrado por Capizzi e Messina [Capizzi 2008, Capizzi and Messina 2008]. Abordagens descentralizadas baseadas em *provisioning* de políticas, porém, também apresentam dificuldades relacionadas à sincronização das políticas no ambiente distribuído.

Este artigo descreve um sistema para gerência de controle de uso no modelo PaaS. Para isso, utiliza uma abordagem híbrida de *provisioning* com gabaritos de regras previamente configurados nos servidores e credenciais de autorização gerenciadas separadamente. As políticas individuais são derivadas localmente, isto é, no servidor onde os mecanismos de avaliação e aplicação se encontram. Isto ocorre através da combinação de informações contidas nas credenciais com os gabaritos de regras instalados no servidor. A proposta elimina a necessidade de sincronização de políticas locais com políticas remotas, presente nos modelos baseados em *provisioning* tradicional, possibilitando a geração de políticas individuais diferenciadas e a melhora da frequência de reavaliação das regras de controle de uso em comparação com as abordagens centralizadas (*outsourcing*).

O artigo está organizado da seguinte maneira: Seção 2 apresenta a fundamentação; A Seção 3 discute a proposta em detalhes; A implementação de um protótipo e testes de avaliação são descritos na Seção 4; Os trabalhos relacionados estão na Seção 5 e a Seção 6 apresenta as conclusões deste trabalho.

2. Fundamentação

Esta seção apresenta uma breve discussão sobre conceitos fundamentais para a compreensão deste trabalho. Nomeadamente, serão abordados os conceitos de plataforma como serviço, arquiteturas de políticas e o modelo de controle de uso.

2.1. Ambientes de Plataforma como Serviço

O ambiente de nuvem pode oferecer modelos de serviço com níveis de abstração que vão do mais baixo (infraestrutura) até o mais alto (software), sendo o modelo intermediário a Plataforma como Serviço (PaaS). De acordo com [Mell 2009], um ambiente de PaaS se caracteriza por abstrair algumas funcionalidades uteis de uso comum para o usuário. Ele pode ser implementado como um ambiente de execução, com bibliotecas que podem ser empregadas pelo usuário para desenvolver aplicações que consomem serviços oferecidos pela plataforma. Os recursos subjacentes são geridos de maneira transparente pela plataforma, isto é, o usuário não tem como gerenciar tais recursos diretamente. É possível, no entanto, controlar a execução de sua aplicação.

Um aspecto característico de ambientes PaaS é a utilização de mecanismos mais leves para isolamento de aplicações. Isto pode ser observado em soluções populares como o Red Hat OpenShift [Red Hat, Inc. 2015], Heroku Cloud Platform [Heroku Inc. 2015] e Cloud Foundry [Pivotal Software Inc. 2015]. Um desses mecanismos é conhecido como virtualização em nível de sistema operacional ou baseada em *containers* [Soltesz et al. 2007]. Trata-se de uma forma menos abrangente de virtualização, para situações em que o custo de virtualização em hardware, como geralmente utilizado no modelo IaaS, é muito alto. Pode ser utilizado em ambientes de computação de alto desempenho, *clusters*, *grid computing*, *web hosting*, etc.

Este tipo de isolamento oferece maior desempenho bruto e maior escalabilidade. Algumas avaliações experimentais mostraram que em algumas situações a sobrecarga de virtualização completa é muito maior do que a baseada em *containers* [Padala et al. 2007, Chaudhary et al. 2008]. Por essa razão, é uma opção popular para a construção de ambientes de PaaS, permitindo a execução de aplicações concorrentes em um mesmo hospedeiro, sem que as aplicações tenham conhecimento umas das outras.

2.2. Arquiteturas de Políticas

As arquiteturas para configuração e avaliação de políticas têm sido tradicionalmente implementadas de acordo com duas abordagens diferentes, que podem ser resumidas como *provisioning* e *outsourcing*:

Provisioning exige configuração das políticas nos mecanismos de controle local. A política fica em um repositório local. O guardião do recurso consulta um ponto de decisão de política (monitor de referência) local para obter uma decisão e fazer sua aplicação (*enforcement*) [Chan et al. 2001]. Sua vantagem é a robustez, pois não tem dependência externa, porém sua desvantagem é a complexidade para manter atualizadas as políticas distribuídas no ambiente.

Outsourcing atua como um modelo cliente e servidor. Um serviço externo, o monitor de referência, responde a pedidos de avaliação de políticas de todos os guardiões de recursos do ambiente. Cada tentativa de acesso exige uma avaliação de política no serviço remoto. O guardião somente aplica a decisão [Durham et al. 2000]. Sua vantagem é simplificar a atualização das políticas e a implementação do guardião. Suas desvantagens são o custo extra de comunicação e sua fragilidade, pois o avaliador de políticas é um ponto único de falha. Além disso, por ser centralizado, possui maior complexidade inerente para obter escalabilidade compatível com ambientes de nuvem computacional.

2.3. Controle de Uso

O modelo UCON$_{ABC}$ [Park and Sandhu 2004] é a reunião de diversas ideias anteriores para o controle de uso de recursos em um modelo formal e coeso. Ele permite a representação de controles como *Digital Rights Management* (DRM), discricionários, mandatórios, baseados em papéis e baseados em confiança (*trust management*).

UCON$_{ABC}$ possui dois conceitos fundamentais: (i) Continuidade, define que os controles definidos na política sejam avaliados e aplicados durante todo o uso do recurso (antes de iniciar/*pre*, durante/*ongoing* e ao fim/*post*); (ii) Mutabilidade, significa que determinados atributos do sujeito ou objeto podem ser alterados pela política de controle durante o uso.

Os controles são categorizados como sendo autorizações (o usuário precisa ter os direitos para a execução de uma ação), condições (avaliações feitas sobre atributos do ambiente, como a hora ou localização geográfica), obrigações (ações externas, que não estão nem em nível de autorização e nem de condições) e atualizações (modificações que se refletem nos atributos de usuário ou objeto).

3. Gerência Distribuída de Controle de Uso em Plataformas como Serviço

Nesta seção é proposta uma infraestrutura para gerenciamento de controle de uso para PaaS, que utiliza gabaritos de regras e credenciais de autorização para derivar políticas que são avaliadas localmente. A abordagem proposta elimina a necessidade de configuração direta de políticas existente em arquiteturas baseadas em *provisioning*, assim como o custo de comunicação presente em arquiteturas baseadas em *outsourcing*.

Os componentes que compõe a infraestrutura podem ser categorizados como sendo de *domínio administrativo* ou de *domínio local*. No nível administrativo estão os serviços que possibilitam a gerência dos atributos de autorização que compõe a credencial de autorização, e a criação dos gabaritos de regras que são previamente configurados nos *domínios locais* (*hosts*) onde as aplicações são executadas. No nível local estão os componentes para validação de credenciais de autorização, derivação da política de controle de uso, avaliação de políticas, e aplicação da decisão gerada no monitor de referência.

As aplicações de usuário são executadas localmente em ambientes isolados com mecanismos do sistema operacional, que permitem contabilização e controle finos, independente do número de processos em execução dentro deste ambiente isolado. Cada ambiente isolado é subordinado a uma política individual de controle de uso, derivada a partir da credencial de autorização apresentada e dos gabaritos de regras preexistentes no ambiente.

A infraestrutura foi pensada para eliminar a necessidade de configuração de políticas para cada aplicação, assim como a consequente necessidade de rastrear essas políticas para que possam ser sincronizadas quando alguma modificação se faz necessária. Os mecanismos de controle de uso locais permitem um tempo de resposta mais curto que em abordagens baseadas em *outsourcing*, além de eliminar o ponto único de falha próprio destas abordagens: falhas ou mudanças em algum *domínio local* não apresentam impactos no resto do ambiente.

3.1. Domínio Administrativo

O domínio administrativo é constituído de quatro serviços: o *Policy Administration Point* (PAP, ou ponto de administração de política), onde são criados os gabaritos de regras; o gerenciador de atributos (GA), utilizado para definir os atributos de autorização aplicáveis a cada usuário; o *Security Token Service* (STS, ou serviço de *token* de segurança), que emite credenciais de autorização; o serviço de controle de entrada (SCE), que previne que uma credencial de autorização seja indevidamente utilizada em múltiplos *domínios locais*. Esses componentes estão ilustrados na Figura 1.

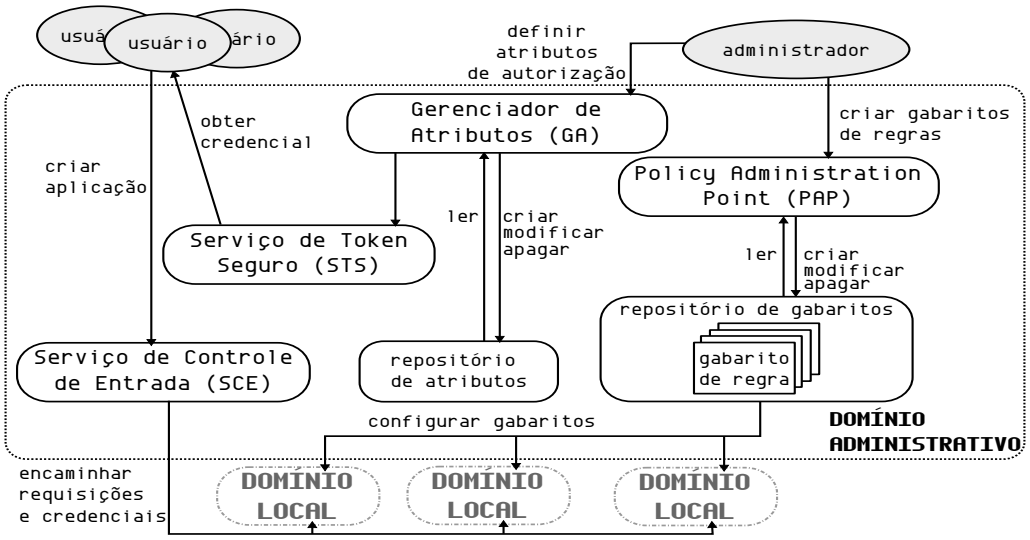


Figura 1. Componentes do Domínio Administrativo

O PAP tem a função de gerenciar um repositório de gabaritos de regras. Cada gabarito refere-se a uma única regra de controle de uso, tendo lacunas que são identificadas por variáveis de nome único (o *id-lacuna*). Esses gabaritos devem representar todas as regras que o administrador deseja impor no ambiente protegido como, por exemplo, regras para controle de horário, utilização de disco, número de instâncias, etc. Além disso, a utilização de gabaritos de regras, ao invés de gabaritos de políticas, se dá pela necessidade de manter a flexibilidade: usuários diferentes podem ser submetidos a combinações diferentes destes gabaritos.

O repositório de gabaritos de regras é configurado em todos os domínios locais – um exemplo concreto consiste em copiar tal repositório na imagem do sistema operacional das máquinas virtuais onde as aplicações serão executadas, de modo que o repositório esteja disponível imediatamente após a inicialização do sistema. O fato dos gabaritos representarem padrões de regras comuns ao ambiente, permite deduzir que atualizações neste repositório serão raras. No entanto, os repositórios contidos nos domínios locais podem ser atualizados a qualquer momento, mediante um sistema de notificação (e.g. quando um gabarito muda no domínio administrativo, ele pode ser difundido para todos os participantes através de uma mensagem de *broadcast* confiável. Os domínios locais atualizam seus repositórios e re-derivam as políticas).

O serviço de GA consiste em um repositório de informações que irão compor a credencial de autorização. Entre as informações gerenciadas está um conjunto de identificadores (chamados *id-lacuna*), que são organizados em subgrupos correspondentes aos

tipos de controles UCON$_{ABC}$ (*pre* e *ongoing*). Esse conjunto irá definir quais gabaritos de regra serão selecionados para derivar a política de controle de uso a ser aplicada. Além disso, o repositório mantém registros das quantidades de recursos que podem ser concedidas a cada usuário, além do montante de recursos que foram concedidos a cada usuário nas credenciais já emitidas. Desta maneira, um usuário pode estar habilitado para uma determinada quantia de recurso (e.g. 100 GB de disco) e ter solicitado apenas uma parcela (e.g. 10 GB), o que permite que esse usuário solicite outras credenciais futuramente. É deste modo que o GA impede que um usuário solicite mais recursos do que aqueles que lhe foram alocados pelo administrador. O conjunto de *id-lacuna*s e os atributos de autorização, formam a credencial que, após ser combinada com os gabaritos de regra da origem a política de controle de uso aplicável a requisição do usuário.

O STS, por sua vez, é o serviço que confecciona a credencial de autorização a pedido do usuário. Para que isso ocorra, o usuário deve enviar um pedido de emissão de credencial ao STS. Após a autenticação do pedido, o STS contacta o GA e solicita a validação das quantidades pedidas pelo usuário: uma solicitação inválida (e.g. por exceder a quantidade permitida) é prontamente recusada. O GA, ao identificar um pedido válido, registra a quantidade solicitada no montante de recursos concedidos ao usuário e retorna uma confirmação ao STS, contendo os atributos solicitados e o conjunto de identificadores de gabaritos de regras aplicável ao usuário. O STS constrói uma credencial contendo estas informações, cifrando a credencial com a chave pública do SCE, de modo que o usuário não saiba quais os detalhes contidos no documento. A credencial gerada é enviada ao usuário que, a partir deste momento, pode invocar a infraestrutura para solicitar seus serviços.

O último serviço do domínio administrativo é o SCE, ou serviço de controle de entrada. Esse componente funciona como o portão de entrada da plataforma, que pode contar com inúmeros *hosts* para a execução de aplicações de usuário, tornando inviável a emissão de credenciais específicas para apenas um *domínio local* (e.g. caso o *host* escolhido esteja com problemas, o usuário só descobrirá isso ao tentar realizar uma requisição, tendo que solicitar a invalidação da credencial atual e a emissão de outra credencial para outro *host*). A emissão de credenciais genéricas de autorização exige, entretanto, que se controle o uso da credencial para que não seja apresentada em dois ou mais lugares simultaneamente. Por estas razões, o SCE tem duas funções: selecionar um *domínio local* em boas condições para atender a requisição do usuário e rastrear onde as credenciais estão sendo usadas, de modo a impedir que tentativas de uso em múltiplos guardiões ocorram.

Os componentes do domínio administrativo possuem menor criticidade em relação ao funcionamento dos domínios locais. Isto é, pequenos momentos de indisponibilidade não irão afetar o funcionamento das aplicações que já estão em execução, pois todos os mecanismos de controle de uso estão no *domínio local*. Contudo, é interessante que os serviços deste nível adotem estratégias para alta disponibilidade, de forma que possíveis problemas neste nível não impeçam que os usuários iniciem novas aplicações ou modifiquem os parâmetros daquelas que já se encontram em execução.

3.2. Domínio Local

O domínio local é composto por todos os mecanismos de controle de uso presentes no sistema local. É nele que a aplicação de usuário é executada em um ambiente isolado,

onde as credenciais de autorização são validadas e combinadas com gabaritos de regras, para gerar a política de controle de uso. A avaliação e aplicação desta política é realizada de modo completamente independente de mecanismos externos. A Figura 2 ilustra os componentes que compõe o mecanismo local. Cada domínio local, presente na Figura 1, consiste de um servidor capaz de executar aplicações de usuário e realizar o controle de uso das mesmas, como será mostrado a seguir.

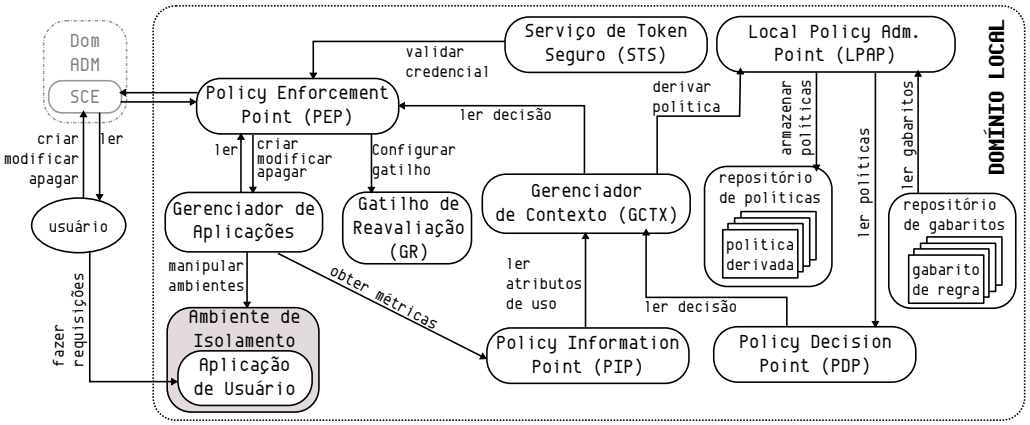


Figura 2. Componentes do Domínio Local

O usuário pode realizar duas modalidades de requisição: de aplicações e de gerenciamento. O primeiro tipo são requisições dependentes do tipo de aplicação que o usuário está executando (e.g. requisições GET para um servidor web). As requisições de gerenciamento são utilizadas para criar, modificar, apagar ou obter informações sobre uma determinada aplicação (e.g. solicitação para executar uma aplicação web). Os mecanismos de controle locais estão focados unicamente no segundo tipo de requisições.

O primeiro item a ser observado na Figura 2 é o serviço de controle de entrada (SCE). Quando o usuário deseja executar uma requisição de gerenciamento, como criar ou modificar uma aplicação, o pedido deve ser encaminhado ao SCE, juntamente com a credencial do usuário. O SCE atua como um despachante de pedidos, externo ao domínio local, que sabe para qual domínio deve encaminhar a requisição de usuário.

O guardião do recurso, conhecido como *Policy Enforcement Point* (PEP), recebe as requisições encaminhadas pelo SCE e deve garantir que estas sejam executadas somente se forem autorizadas. Para que isso seja possível, o primeiro passo a ser realizado consiste na validação da credencial do usuário.

A validação é realizada no serviço de *token* de segurança (STS). Nesse estágio são verificadas as datas de validade, autenticidade das assinaturas, relações de confiança, integridade dos dados, entre outros. Qualquer falha na validação da credencial faz com que a requisição do usuário seja considerada inválida.

Considerando que o PEP obtenha uma resposta positiva do STS, o próximo passo é submeter uma requisição de autorização ao gerenciador de contexto (GCTX) juntamente com informações contextuais (e.g. credencial do usuário, requisição solicitada, argumentos de requisição, etc). O PEP fica aguardando uma resposta do GCTX com a decisão de autorização a ser aplicada.

Procedimento 1 Derivação de Política

1: $PS \leftarrow grupos\_de\_politicas(credencial)$
2: $DS \leftarrow cria\_novo\_agrupamento()$
3: **enquanto** $PS \neq \emptyset$ **faça**
4: $P \leftarrow retira\_proxima\_politica(PS)$
5: $D \leftarrow cria\_nova\_politica(P)$
6: $IS \leftarrow ids\_de\_gabarito(P)$
7: **enquanto** $IS \neq \emptyset$ **faça**
8: $ID \leftarrow retira\_proximo\_id(IS)$
9: $GI \leftarrow busca\_gabarito(ID, repositorio)$
10: **se** $GI = \emptyset$ **então**
11: **retorna** falha
12: **fim se**
13: $configura(GI, credencial)$
14: $adiciona(GI, D)$
15: **fim enquanto**
16: $adiciona(D, DS)$
17: **fim enquanto**
18: $salva(DS, repositorio)$
19: **retorna** sucesso

O gerenciador de contexto é o componente que atua como integrador das partes do domínio local. Sua primeira tarefa é interpretar o formato de comunicação utilizado pelo PEP, convertendo-o para um formato útil aos demais componentes. Após extrair a credencial de política da requisição enviada pelo PEP, o GCTX envia uma requisição para derivação de política ao gerenciador de políticas local, ou *local policy administration point* (LPAP). Em paralelo o GCTX recupera quaisquer informações de utilização de recursos associadas ao usuário, aplicação (caso já esteja em execução) e ambiente através do repositório de informações (*Policy Information Point*, PIP).

O LPAP tem a função de confeccionar políticas de controle de uso derivadas da credencial de usuário e das regras contidas no repositório de gabaritos. O Procedimento 1 resume as ações executadas pelo LPAP para derivar uma política. O LPAP identifica quais grupos de políticas devem ser criados (e.g. *pre-authorization*, *ongoing-conditions*, etc) e guarda em PS. As políticas que serão criadas são agrupadas em um único documento, DS. O LPAP cria um novo documento de política para cada grupo de PS e guarda em D. Os gabaritos que devem ser aplicados são descobertos através da inspeção da lista de identificadores contida em cada entrada P. O LPAP recupera, uma a um, os gabaritos correspondentes e executa um processo de configuração dos campos vazios, devidamente identificados, com as informações contidas na credencial do usuário. Após configurar todos os gabaritos, o documento da política é adicionado ao agrupamento DS (linha 16). Quando não houverem mais políticas em PS, a política resultante (contida em DS) é armazenada no repositório de políticas. Uma mensagem de sucesso é retornada ao GCTX, permitindo que o processo de avaliação de autorização continue.

O PIP, por outro lado, é o componente responsável por disponibilizar informações de uso ao GCTX, através de uma interface unificada. As informações de uso são coletadas

por outros componentes como, por exemplo, o gerenciador de aplicações. Os agentes de coleta de métricas utilizam as interfaces nativas do sistema operacional para contabilizar o consumo realizado por aplicações individuais, assim como o estado atual do sistema (e.g. o *load average*, número de aplicações em execução, etc). Esses dados são essenciais para a correta avaliação das políticas de controle de uso.

Procedimento 2 Avaliação de Política

1: $S \leftarrow sujeito(credencial), O \leftarrow objeto(credencial)$
2: $C \leftarrow contexto(credencial), E \leftarrow endereco(LPAP)$
3: $PS \leftarrow recupera\_politica(S, O, E)$
4: **se** $PS = \emptyset$ **então**
5: **retorna** falha
6: **fim se**
7: **enquanto** $PS \neq \emptyset$ **faça**
8: $P \leftarrow retira\_proxima\_politica(PS)$
9: **enquanto** $P \neq \emptyset$ **faça**
10: $R \leftarrow retira\_proximo\_regra(P)$
11: **se** $avalia(R, S, O, C) = Negado$ **então**
12: **retorna** falha
13: **fim se**
14: **fim enquanto**
15: **fim enquanto**
16: **retorna** sucesso e gatilho de reavaliação

Antes de solicitar a avaliação de política, GCTX inspeciona a requisição para verificar se ela se aplica a uma aplicação já em execução ou se trata-se de uma nova instância. Isso é essencial para que a fase correta do controle de uso seja aplicada. Como os ambientes de isolamento representam o objeto a ser controlado, sua existência implica uma sessão de uso já em execução. O GCTX utiliza essa informação para solicitar uma decisão ao mecanismo de decisão local (*Policy Decision Point*, PDP) e fica aguardando pela resposta.

O PDP é o componente especializado em interpretar as regras de políticas e verificar se as regras são atendidas com a credencial apresentada pelo usuário dentro das condições estabelecidas, ver Procedimento 2. Para que possa fazer isso, o PDP deve recuperar a política aplicável ao usuário e aplicação desejada. Essa política é obtida através do LPAP, que utiliza o identificador do usuário e aplicação para recuperar o documento apropriado no repositório de políticas. A ausência de políticas faz com que a requisição seja automaticamente negada (por este motivo o GCTX deve aguardar a confirmação LPAP após a derivação da política). De posse da política o PDP combina as informações contextuais enviadas pelo GCTX e verifica se a política está sendo respeitada. A decisão obtida é retornada ao GCTX juntamente com os detalhes de aplicação da mesma.

O GCTX converte a resposta obtida do PDP para um formato compreensível pelo PEP. Nessa resposta se encontram a decisão que deve ser aplicada (autorizado ou negado) e o gatilho de reavaliação da política, isto é, a frequência com que se deve reavaliar a política. O PEP configura o gatilho de reavaliação (GR) no componente que recebe o mesmo nome. O GR funciona como um alarme que informa ao PEP quando uma política

deve ser reavaliada. Quando o gatilho é criado, o PEP repassa os dados da requisição sendo que, no momento da reavaliação, o PEP terá informações referentes ao reinício do processo de reavaliação. O PEP encaminha requisições autorizadas para o gerenciador de aplicações local, que fará a execução efetiva da requisição do usuário. A resposta obtida nesse estágio é retornada ao usuário (e.g. informações de acesso ip:porta).

Como o modelo proposto é independente da tecnologia escolhida, o gerenciador de aplicações implementa as ações necessárias para realizar o pedido do usuário. Resumidamente, é criado um novo ambiente isolado no sistema operacional, o limite de CPU e disco é configurado de acordo com os parâmetros recebidos, assim como os detalhes de acesso a rede. O gerenciador instala a aplicação no ambiente isolado e a executa. Se não encontrar nenhum erro, os dados de acesso são retornados, caso contrário uma mensagem de erro é retornada.

3.3. Gabaritos, Derivação e Atualizações

Cada gabarito de regra representa um controle bem definido. Na Figura 3 apresentamos uma versão simplificada de um gabarito para controle de tempo de processamento (CPU). O primeiro item a ser observado é o identificador *RuleID*: este valor identifica o gabarito unicamente, sendo que a credencial do usuário carrega esse valor como um de seus atributos. O segundo aspecto importante é o identificador *TotalCpuTime*: é o parâmetro que deve ser configurado para cada usuário, isto é, o *id-lacuna* discutido anteriormente. Quando o usuário solicita a emissão de uma credencial, ela conterá um atributo com o identificador *TotalCpuTime* indicando o limite de uso de CPU. O terceiro aspecto é a variável *usedCpu*, este atributo não é modificado com dados da credencial, pois é obtido localmente através do PIP – é a informação de contabilidade local.

```
1 <Rule RuleId="CPURule" Effect="Permit">
2   <Target><Any/></Target>
3   <Condition>
4     <Apply FunctionId="integer-less-than-or-equal">
5       <Apply FunctionId="integer-one-and-only">
6         <AttributeDesignator Category="access-subject"
7         AttributeId="usedCpu" DataType="integer"/>
8       </Apply>
9       <Apply FunctionId="integer-one-and-only">
10        <AttributeValue DataType="integer">
11          ${TotalCpuTime}
12        </AttributeValue>
13      </Apply>
14    </Apply>
15  </Condition>
16 </Rule>
```

Figura 3. Exemplo de Gabarito de Regra

Cada credencial carrega os identificadores de gabaritos separados por grupos (e.g. grupos *pre* e *ongoing*). Os atributos de configuração são carregados em uma seção separada. Como exibido na Figura 3, a derivação da regra consiste em recuperar o gabarito correto (identificado pelo *RuleID*) e substituir os campos (identificados por ${}) com o valor dos atributos correspondentes das credenciais. As regras resultantes são organizadas em políticas separadas, de acordo com a fase de aplicação, e armazenadas em um documento agrupador (*PolicySet*). Para permitir a individualização, os *PolicySets* e suas políticas, possuem um alvo (*Target*) aplicável somente ao usuário e a aplicação da requisição (identificada por um ID único).

Ao salvar as políticas derivadas no repositório, o LPAP registra os *metadados* da política, que inclui, entre outras informações, os identificadores dos gabaritos que deram origem. Desta maneira, supondo que um ou mais gabaritos sejam modificados pelo administrador, o LPAP tem como identificar quais políticas dependiam daquele gabarito e desencadear a sua re-derivação a partir dos gabaritos atualizados.

4. Implementação e Avaliação

O sistema proposto foi avaliado através da implementação de um protótipo dos componentes utilizados no domínio local. Isto permitiu verificar se os algoritmos locais são capazes de implementar os mecanismos de controle de uso desejados. Além disso, foi possível fazer uma análise de desempenho do avaliador de políticas (PDP) e identificar a diferença do tamanho das mensagens transferidas na proposta com mensagens que seriam utilizadas em abordagens baseada em *provisioning* puro e *outsourcing*.

4.1. Implementação do Protótipo

Para a implementação do protótipo foram utilizados alguns projetos de código livre, que foram integrados através da linguagem de programação Java. A compartimentalização de aplicações de usuários é realizada através das *jails* do sistema FreeBSD [The FreeBSD Project 2015], que oferecem os mecanismos para monitorar o consumo de recursos da aplicação e para gerenciar a execução da mesma. As credenciais de autorização seguem a especificação SAML [OASIS 2014a], usando mensagens do tipo *AttributeStatement* que são criadas e manipuladas através do projeto OpenSAML [Internet2 2015]. As políticas de controle de acesso seguem o padrão XACML [OASIS 2014b] e são avaliadas através da biblioteca WSO2 Balana [WSO2 Inc. 2015].

Os identificadores de gabaritos de regras que deverão ser aplicados ao usuário, são transportados como se fossem um atributo da mensagem SAML. São construídos como um objeto XML que imita a estrutura hierárquica da política que deverá ser derivada no domínio local – cada fase da sessão de uso (*pre* ou *ongoing*) possui sua própria lista de identificadores, na ordem que devem ser configurados, assim como demais informações relevantes (e.g. qual o algoritmo de combinação de regras a ser usado). Essas mensagens são protegidas criptograficamente, garantindo sua autenticidade e integridade.

O domínio local, recebe as requisições contendo a credencial de autorização em um serviço web estilo *REST*, invoca um processo gerenciador de contexto feito em java, com módulos para a validação da credencial, derivação da política XACML e sua posterior avaliação. Uma *thread* é configurada com um parâmetro contido no resultado da avaliação, para periodicamente solicitar a reavaliação da política. Os dados de sessão e políticas são mantidos em um diretório local legível apenas pelo gerenciador de contexto.

4.2. Avaliação

O tempo de resposta do monitor de referência (PDP) em face de um número crescente de políticas armazenadas localmente foi medido (Figura 4, lado direito) e considerado baixo, em média, sempre permanecendo abaixo de 2 ms. Com poucas políticas (menos que 1000), os piores tempos foram maiores que nos demais cenários, ainda assim ficaram dentro de um limite praticável de 10 milissegundos.

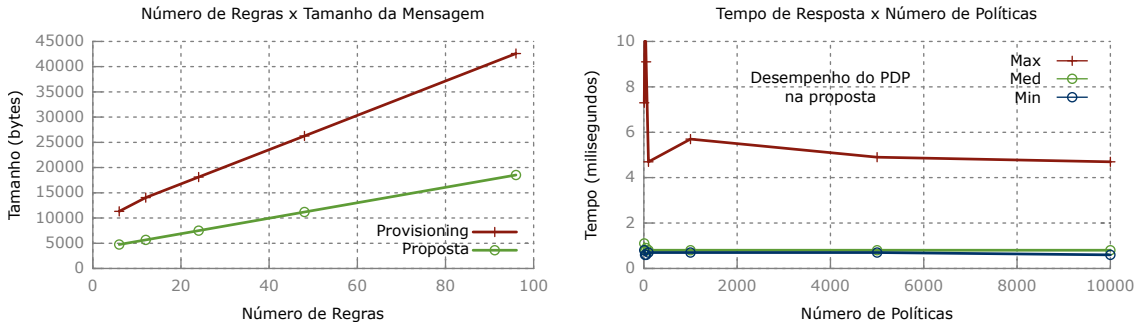


Figura 4. Avaliação: Provisioning *versus* Proposta e desempenho do PDP em relação ao número de políticas

O segundo teste (Figura 4, lado esquerdo) demonstra a vantagem de se utilizar uma abordagem híbrida com gabaritos ao invés de configuração de políticas. Para a utilização do mesmo número de regras (controles) a abordagem de *provisioning* se mostra computacionalmente mais cara. O modelo híbrido utilizou mensagens na faixa de 4754 a 18524 bytes, enquanto o modelo *provisioning* utilizou de 11314 a 42594 bytes, num cenário envolvendo de 6 a 96 regras.

Os testes mostraram as vantagens da proposta em relação aos modelos tradicionais de gerência de políticas (*provisioning* e *outsourcing*), sendo que os tempos de resposta são melhores e a verbosidade das mensagens é menor. É importante ressaltar que o modelo *outsourcing* exige um protocolo do tipo requisição e resposta, que além de mais lento pode gerar uma significativa quantidade de mensagens na rede.

5. Trabalhos Relacionados

A proposta desenvolvida em [Marcon Jr. et al. 2014] descreve uma arquitetura para controle de uso resiliente para computação em nuvem. O trabalho adota uma arquitetura do tipo *outsourcing* para permitir o controle de uso em um ambiente distribuído. Os autores propõem a utilização de espaços de tuplas para lidar com a alta demanda vinda dos sistemas hospedeiros, deste modo, os mecanismos de avaliação de políticas e contabilização de uso lançam novas instâncias conforme a necessidade. A resiliência da proposta consiste em permitir que discrepâncias na utilização de recursos, entre reavaliações de políticas, sejam acomodadas através de uma quota de recurso que funciona como uma margem de segurança. Entre as principais diferenças entre aquele trabalho e esta proposta estão: arquiteturas diferenciadas (híbrida de *provisioning* com credenciais de autorização e gabaritos de regras *versus outsourcing* puro e políticas estáticas); escopos diferentes (controle local *versus* controle de alto nível) e reavaliação de políticas (parametrizável *versus* fixa).

Em [Lazouski et al. 2012] foi descrita uma arquitetura para controle de nuvem em um ambiente IaaS. Os autores optaram por realizar extensões à linguagem XACML, com a finalidade de expressar atualização de atributos e fase de reavaliação. A arquitetura funciona em modo *outsourcing*, com um componente que monitora modificações em atributos de uso, com periodicidade pré-definida, e dispara a reavaliação de políticas de controle de uso. A possível sobrecarga dos mecanismos de avaliação de política não é abordada. Nossa proposta difere-se tanto pela abordagem arquitetural, que é híbrida, quanto pela ausência de extensões a linguagem XACML, além de oferecer reavaliação pa-

rametrizável das políticas, escopo do controle local e granularidade mais fina(aplicações *versus* máquinas virtuais).

Uma arquitetura de controle de acesso baseada no modelo UCON$_{ABC}$ é apresentada em [Danwei Chen 2009]. Sua contribuição é a inclusão de um modelo de negociação, acoplado a arquitetura de autorização, flexibilizando o controle de acesso em nuvens computacionais. O sujeito tem a possibilidade de obter outra opção de acesso através de negociação, em certas situações, ao invés de ser prontamente rejeitado. O trabalho encontra-se apenas como proposta conceitual, sem demonstração de viabilidade para ambientes de nuvem ou computação distribuída.

6. Conclusão

Este artigo apresentou e avaliou uma proposta de arquitetura para gerência distribuída de controle de uso em ambientes de PaaS, tendo como características distintivas a utilização de gabaritos de regras para derivar políticas de controle de acesso com base em credenciais de autorização. Os gabaritos são previamente armazenados nos sistemas locais, onde a avaliação da política e *enforcement* são realizados. As credenciais evitam o transporte de políticas completas através da rede, com consequente redução no tamanho das mensagens, sem contudo perder a flexibilidade na definição de políticas individuais. A sincronização de políticas é obtida através do envio de credenciais SAML com regras individuais (quando o domínio local detecta a mudança dos valores nas assertivas de autorização individuais, as políticas locais afetadas são regeradas e reavaliadas). Os testes realizados indicam que a proposta é viável, pois reduz o custo de comunicação existente em abordagens tradicionais enquanto obtém os benefícios próprios de uma abordagem de *provisioning* sem, contudo, exibir a complexidade para manter as políticas sincronizadas.

Futuramente espera-se desenvolver uma arquitetura descentralizada de compartilhamento de atributos entre domínios locais para reconfiguração dinâmica dos atributos de autorização. Isso viabilizará a cooperação do domínios locais, otimizando o uso de recursos, sem a necessidade de um sistema centralizado que exige mecanismos complexos para obter escalabilidade e robustez na presença de faltas.

Referências

Capizzi, S. (2008). *A Tuple Space Implementation for Large-scale Infrastructures*. Tese de Doutorado, Università di Bologna.

Capizzi, S. and Messina, A. (2008). A Tuple Space Service for Large Scale Infrastructures. In *IEEE 17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 182–187.

Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and Smith, A. (2001). COPS usage for policy provisioning (COPS-PR). RFC 3084.

Chaudhary, V., Cha, M., Walters, J., Guercio, S., and Gallo, S. (2008). A comparison of virtualization technologies for HPC. In *IEEE 22nd International Conference on Advanced Information Networking and Applications, AINA.*, pages 861–868.

Danwei Chen, Xiuli Huang, X. R. (2009). Access Control of Cloud Service Based on UCON. In *Cloud Computing*, pages 559–564. Springer.

Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R., and Sastry, A. (2000). The COPS (common open policy service) protocol. RFC 2748.

Hayes, B. (2008). Cloud computing. *Communications of the ACM*, 51(7).

Heroku Inc. (2015). Heroku. `https://www.heroku.com/`. Acessado: 30-03-2015.

Internet2 (2015). Opensaml 2 for java. `https://wiki.shibboleth.net/confluence/display/OpenSAML/Home`. Acessado: 30-03-2015.

Lazouski, A., Mancini, G., Martinelli, F., and Mori, P. (2012). Usage Control in Cloud Systems. In *IEEE International Conference for Internet Technology And Secured Transactions*, pages 202–207.

Marcon Jr., A. L., Santin, A. O., Stihler, M., and Bachtold, J. (2014). A UCON$_{ABC}$ Resilient Authorization Evaluation for Cloud Computing. *IEEE Trans. Parallel Distrib. Syst.*, 25(2):457–467.

Mell, Peter e Grance, T. (2009). The NIST Definition of Cloud Computing. *National Institute of Standards and Technology*, 53(6):50.

OASIS (2014a). Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`. Acessado: 30-03-2015.

OASIS (2014b). eXtensible Access Control Markup Language (XACML) Version 3.0. `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html`. Acessado: 30-03-2015.

Padala, P., Zhu, X., Wang, Z., Singhal, S., and Shin, K. G. (2007). Performance Evaluation of Virtualization Technologies for Server Consolidation. *HP Labs Tec. Report*.

Park, J. and Sandhu, R. (2004). The UCONabc Usage Control Model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174.

Pivotal Software Inc. (2015). Cloud Foundry. `http://cloudfoundry.org/index.html`. Acessado: 30-03-2015.

Red Hat, Inc. (2015). Openshift. `https://www.openshift.com/`. Acessado: 30-03-2015.

Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. (2007). Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3):275–287.

Takabi, H., Joshi, J. B., and Ahn, G.-J. (2010). Security and Privacy Challenges in Cloud Computing Environments. *IEEE Security & Privacy*, 8(6):24–31.

The FreeBSD Project (2015). The FreeBSD Project. `https://www.freebsd.org`. Acessado: 30-03-2015.

WSO2 Inc. (2015). Balana XACML for Authorization. `https://svn.wso2.org/repos/wso2/trunk/commons/balana/`. Acessado: 30-03-2015.

PATROCÍNIO DIAMANTE

cgi.br
Comitê Gestor da Internet no Brasil

nic.br
Núcleo de Informação e Coordenação do Ponto BR

Microsoft Research

PATROCÍNIO OURO

CAPES

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

Dinâmica Telecom

PATROCÍNIO PRATA

FAPES
FUNDAÇÃO DE AMPARO À PESQUISA DO ESPÍRITO SANTO

PATROCÍNIO BRONZE

Google

EMC²

Padtec

COLABORAÇÃO

INSTITUTO FEDERAL
ESPÍRITO SANTO
Campus Serra

Ruckus
Simply Better Wireless.

PoP-ES
Ponto de Presença da RNP no Espírito Santo

RNP

PROMOÇÃO

SBC
Sociedade Brasileira de Computação

LARC

REALIZAÇÃO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DOCETE OMNES GENTES

UFES

PRODUÇÃO

ateliê
DE EVENTOS